Università degli Studi di Padova

FACOLTÀ DI INGEGNERIA

Corso di laurea in Ingegneria Informatica

PROGETTAZIONE E SVILUPPO DI UN SISTEMA PER LA GESTIONE DI UNA RETE DI ILLUMINAZIONE PUBBLICA

RELATORI: Prof.ri Massimo Rumor e Sergio Congiu

LAUREANDO: Luca Guadagnin

Matricola 581195

Anno Accademico 2010/2011

INDICE

1. INTRODUZIONE

Il tirocinio si è svolto presso il CED del Comune di Feltre (BL) dal 5/7/2010 al 19/11/2010, per un totale di 500 ore.

L’incarico assegnato è stato quello di seguire la realizzazione di un sistema che potesse censire tutta la struttura dei punti luce, dislocata sull’intero territorio comunale.

Il tutto è avvenuto in un contesto organizzativo sempre più rivolto, nei vari ambiti di competenza, alla realizzazione di un Sistema Informativo che integri i dati con le informazioni cartografiche.

Questa concomitanza ha assunto particolare rilevanza poiché il DB contenente i dati sui punti luce, successivamente all'implementazione, si sarebbe dovuto integrare con il contesto appena descritto.

2. DESCRIZIONE DELL'ENTE

I fabbisogni informativi del Comune di Feltre sono vari e il compito del Ced sta proprio nel mettere in piedi e mantenere i sistemi necessari per raggiungere i vari scopi.

In tal senso, le attività dell'ufficio appena menzionato sono abbastanza articolate e possono essere classificate nelle seguenti categorie:

- *Attività responsabile di procedura*. Questo ruolo consiste: nel monitorare se i flussi informativi presenti nella rete sono conformi alle necessità; in tutta quell'attività di supporto fornita per dare quelle conoscenze specialistiche necessarie, per esempio, all'estrazione di dati non reperibili mediante l'uso di funzioni già definite. Tali attività riguardano l'area amministrativa (protocollo, rilevazione presenze), quella finanziaria (contabilità, inventario, tributi), dei servizi sociali e demografici (anagrafe, materia elettorale, istruzione, cultura), pianificazione e gestione del territorio (urbanistica, progettazione, patrimonio comunale, pratiche edilizie);
- *Attività di Infocenter*. Con questa espressione si indicano quei processi volti da una parte all'assistenza interna ai dipendenti dei vari uffici, analizzando l'esigenze degli utenti e cercando di adattare i prodotti alle necessità riscontrate, e dall'altra alla divulgazione di informazioni alla cittadinanza, tramite i vari portali adottati (come MyPortal). L'attuazione di tutto ciò si ha nella gestione degli aggiornamenti SW dei vari PC e delle procedure volte a fornire dei servizi all'esterno (ICI, autocertificazioni, accesso agli atti amministrativi);
- *Periodica analisi del mercato*, con lo scopo di tenere sott'occhio tutti i prodotti che possono apportare miglioramenti al Sistema Informativo già in uso. Questa attività ha assunto maggiore importanza negli ultimi anni, cioè da quando il Comune di Feltre, in sintonia con l'intero mondo della pubblica amministrazione, ha cominciato a guardare con crescente attenzione alla possibilità di rimpiazzare ciò che già aveva con nuovi sistemi Open Source;
- *Gestione del sistema*. Da quando gli enti pubblici, come le aziende, hanno cominciato a dotarsi di reti interne per collegare fra loro tutte le postazioni informatiche, i benefici raggiunti sono stati molti ma, allo stesso tempo, sono sorte anche nuove problematiche. Tutto ciò ha fatto si che si affermasse un tipo di attività rivolta alla gestione delle apparecchiature HW e delle autorizzazioni di accesso ai dati. Questa parte si svolge tramite la configurazione degli elaboratori

centrali, l'amministrazione della rete di trasmissione dati, la gestione della sicurezza, la valutazione sull'acquisizione di HW e la verifica su come viene gestita l'assistenza fornita in outsourcing;

- *Gestione amministrativa*, svolta seguendo tutte le pratiche collegate alle forniture di materiale HW e SW di cui necessita il Sistema Informativo comunale, sia che si tratti della necessità di un specifico ufficio o dell'intera amministrazione. Ciò obbliga a seguire, passo passo, tutte le gare e i contratti di fornitura, manutenzione e assistenza riferiti alla apparecchiature informatiche;
- *Gestione operativa*, di vitale importanza per la messa in esecuzione di tutte le elaborazioni batch richieste. In tal senso, assume grande importanza la pianificazione di tutte le procedure definite per l'esecuzione di copie di sicurezza, fondamentali per ripristinare le situazioni precedenti ad eventuali crash.

Tale supporto viene dato dal Ced anche a coloro che si occupano della gestione dell'intera rete di illuminazione pubblica.
In particolare, il contesto organizzativo incaricato di seguire questa materia si articola sostanzialmente in 2 parti:

- L'ufficio *Lavori Pubblici*, che si occupa di tutta la parte decisionale, che va dalla progettazione di nuove tratte di punti luce agli accordi con i fornitori di energia elettrica;
- L'ufficio *Manutenzioni*, che è incaricato, in un momento successivo alla messa in opera degli interventi, di seguire tutta la parte sul funzionamento della rete, intervenendo in caso di guasti di qualsiasi natura, dalla semplice sostituzione di una lampada fulminata alla riparazione di una linea che ha subito danni ai cavi dell'alimentazione.

Gli aspetti che tale amministrazione pubblica si trova a dover gestire in merito a questa tematica sono molteplici, fra i quali:

- *localizzazione della varie componenti* che formano la rete di illuminazione, necessario al fine di poter indicare la zona dove l'elemento è dislocato;
- *gli aspetti contrattuali* legati ad ogni singolo quadro elettrico, tali da rendere possibile una conoscenza rapida del fornitore (come in caso di guasti);
- *le vie alimentate da ogni apparato*, per poter sapere le zone che rimangono al buio

a causa di un guasto tecnico su un certo quadro;

- *la natura delle linee elettriche* che permettono il collegamento di ciascun punto luce ad un certo quadro, per sapere quali sono le possibilità di modifica sulla rete offerte dalle infrastrutture a disposizione;
- *la tipologia di lampade* assegnate a ciascun punto luce, che permette di apprendere, senza troppe difficoltà, la lampada che si deve avere a disposizione per effettuare la sostituzione di quella esaurita.

Gli strumenti tecnologici che vengono attualmente utilizzati per soddisfare questo tipo di esigenze sono essenzialmente 2:

- *AutoCAD Map* per la gestione di tutto ciò che riguarda la parte cartografica, integrata con informazioni associate ad ogni elemento censito (es: codici identificativi);
- una serie di *fogli elettronici*, uno per ogni quadro elettrico, dove vengono riportati, nel dettaglio, tutti i dati riguardanti i vari elementi facenti parte della rete di illuminazione.

2.1 AUTOCAD MAP

AutoCAD Map non è altro che un'estensione di *AutoCAD* che nasce con lo scopo di poter elaborare vari aspetti riguardanti la cartografia. In particolare, permette l'interazione tra ambienti CAD e GIS, a prescindere che si abbia a che fare con tecnologia *vettoriale* (cioè rappresentando le mappe mediante la definizione di punti, linee, curve e poligoni) o con quella *raster* (dove viene utilizzato il concetto di pixel come elemento fondamentale dell'immagine).

Tutto ciò fornisce l'opportunità di gestire i vari aspetti riguardanti la cartografia avvalendosi del formato *DWG*, appartenente all'ambiente AutoCAD, con la possibilità di importare ed esportare le mappe nei vari formati GIS che, al contrario del DWG, sono open source.

Quanto detto assume ancora più importanza considerando la tecnologia *FDO* (Feature Data Objects, cioè Fonti Dati Oggetti cartografici), introdotta nel 2007, che ha permesso l'introduzione del seguente meccanismo:

- accesso immediato ai dati presenti in formato GIS, senza necessità di alcun tipo di importazione;
- la modifica di tali dati avvalendosi degli strumenti messi a disposizione da AutoCAD.

Va specificato che AutoCAD Map, anche prima dell'introduzione di questo meccanismo, era in grado di gestire formati alternativi al DWG, come *Shapefile*, uno degli standard dell'ambiente GIS. Questo vuol dire che il vantaggio introdotto dalla FDO non è tanto quello della semplice interazione fra 2 ambienti differenti, già svolta come indicato all'inizio di questo paragrafo, ma nel fatto che è completamente eliminata la fase in cui venivano applicate le tecniche di importazione – esportazione, risparmiando una quantità significativa di tempo.
Ciò è concretamente realizzato tramite la connessione da parte di Map verso quelli che vengono chiamati *Provider FDO*, che consentono di accedere ad archivi di dati di varia natura, fra i quali:

- file vettoriali (.shp);
- file raster (.tiff, .jpg);
- database (come MySQL);
- geodatabase (come PostgreSQL con estensione PostGIS);
- server cartografici da cui poter attingere le mappe.

Questo approccio "elastico" nel trattare l'informazione cartografica da la possibilità, per esempio, di sovrapporre i dati geospaziali derivanti da un Provider FDO con dati propri dell'ambiente AutoCAD e creare delle etichette che permettano di visualizzare i dati riferiti ad un determinato oggetto.
Questa modalità di operare, sovrapponendo layer derivanti da fonti differenti, assume un significato ancora più ampio considerando il gran numero di sistemi di coordinate esistenti nella realtà e messe a disposizione da Map. Infatti, questa estensione di AutoCAD supporta la conversione da un sistema all'altro, grazie ad apposite query che proiettano un DWG da un tipo di coordinate all'altro. A titolo puramente indicativo, questo aspetto può diventare di chiara utilità quando, per esempio, ci si

trova nella situazione dove ad un layer catastale, basato su sistema Cassini-Soldner, debba essere sovrapposto un altro layer contenente la disposizione delle piazzole ecologiche, individuate con coordinate Gauss-Boaga.

A questo punto è necessario discutere su come un ambiente AutoCAD, nato su una logica totalmente vettoriale, possa riuscire a gestire anche immagini in cui la tecnica utilizzata è quella raster, come nel caso delle ortofoto. Questo è divenuto possibile con la realizzazione dell'estensione *Raster Design*, in cui l'obiettivo è stato quello di fornire uno strumento adatto a trattare questo tipo di immagini, in modo tale da integrarle e modificarle direttamente all'interno di AutoCAD Map. L'importanza di questo strumento si fonda su 2 aspetti:

- la *vettorizzazione* di un'immagine raster;
- l'opportunità, ancora più rilevante della precedente, di *manipolare direttamente* i raster come fossero oggetti nativi di AutoCAD, senza che sia necessaria alcuna conversione preventiva.

Tutto ciò diventa importante nel momento in cui si ha bisogno di raffinare i dati a disposizione, come nel caso in cui si voglia affiancare ad una qualsiasi informazione memorizzata in forma vettoriale una foto satellitare, che va georeferenziata adeguatamente.

Va comunque specificato che questi tipi di conversione, ottenibili tramite una serie di "deformazioni", sono eseguibili, in modalità non automatica, anche solamente utilizzando AutoCAD Map. Il vero valore aggiunto derivante dall'utilizzo combinato di AutoCAD Map e Raster Design sta nel fatto di poter ricondurre i dati provenienti dalle varie sorgenti ad un unico sistema di riferimento. Infatti, mentre con il primo si può ricondurre il tutto allo stesso sistema di coordinate, con il secondo si riesce a rendere compatibile un ambiente basato sulla logica vettoriale con le immagini raster. Ciò è di vitale importanza per un ente pubblico, come il Comune di Feltre, dove sono frequenti le situazioni in cui, all'interno di processi di digitalizzazione dell'informazione, ci si trova nella condizione di dover convertire dei dati catastali in forme che rispondano meglio alla realtà. Un esempio può essere quello di passare da un sistema di coordinate Cassini-Soldner, usato nelle mappe catastali, ad un sistema più attento al fatto che i riferimenti siano presi tenendo conto anche della

conformazione del territorio (come le pendenze). Inoltre, nel caso se ne verifichi la necessità, il tutto può essere sovrapposto a delle ortofoto, che sono notoriamente dei raster.

L'ultimo aspetto che va menzionato è quello che si occupa delle possibili imprecisioni derivanti dall'importazione di alcune informazioni nel formato DWG. Infatti, per la natura stessa del formato, è possibile trovarsi a riscontrare particolari che possono "difettare" il disegno, come contorni non chiusi o vertici che non combaciano. Per questo motivo, in Map sono state previste delle procedure il cui compito è proprio quello di eliminare tali difetti.

Ricapitolando brevemente i vantaggi offerti da AutoCAD Map, si può dire che presenta:

- facilità d'uso per chi conosce bene AutoCAD, come coloro che lavorano all'ufficio Lavori Pubblici;
- possibilità di lavorare sia con CAD che con GIS;
- disponibilità di interessanti opzioni di stampa fornite da AutoCAD;
- opportunità di pubblicare direttamente sul Web, grazie alla tecnologia FDO e ad un server cartografico;
- gestione delle immagini in formato Raster, mediante all'estensione Raster Design;
- conversione tra sistemi di coordinate differenti.

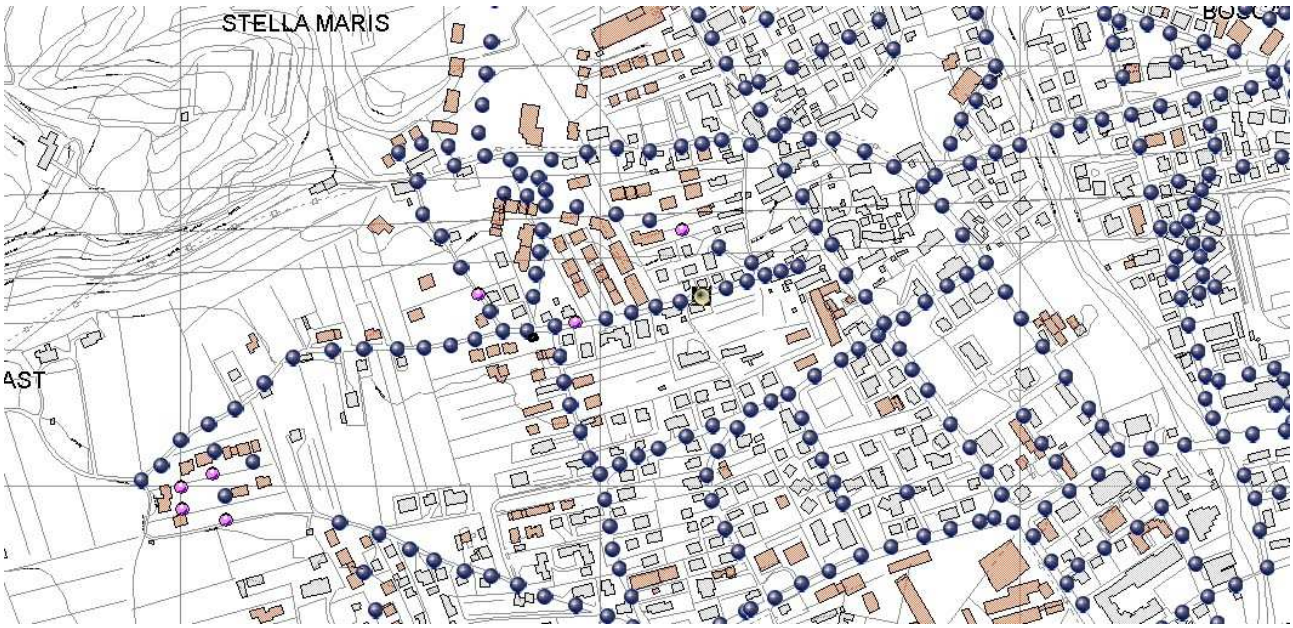


Figura 2.1 – Layer contenente la disposizione dei punti luce del Comune di Feltre

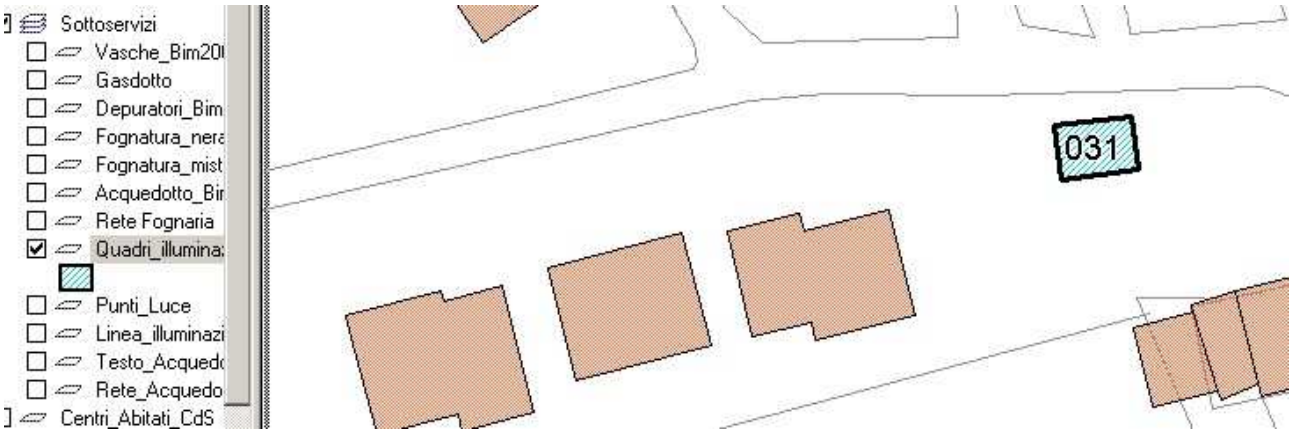
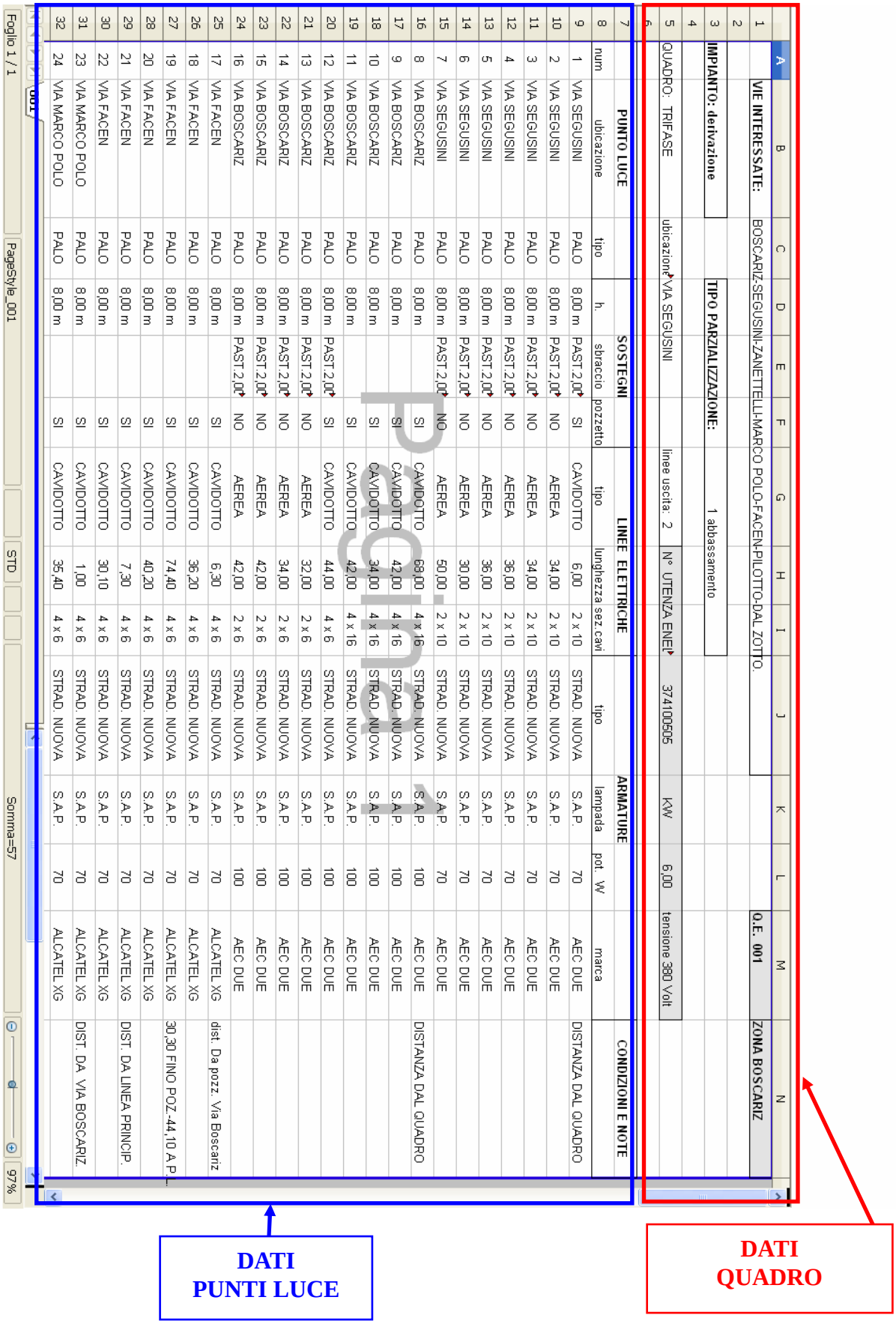


Figura 2.2 – Layer dei quadri elettrici che alimentano i punti luce sopra illustrati

2.2 FOGLI ELETTRONICI

Figura 2.3 – esempio dell'attuale modalità di catalogazione dei dati, tramite una griglia fatta con Excel



3. DESCRIZIONE DEL PROGETTO

3.1 OBIETTIVI

Nella descrizione dell'ente, fatta nel paragrafo precedente, sono stati presentati degli aspetti che, nonostante l'utilizzo di strumenti abbastanza evoluti (come AutoCAD Map), mostrano alcune criticità che l'intervento ha lo scopo di rimuovere. Infatti, nel momento in cui un argomento come quello della gestione della rete di illuminazione coinvolge più uffici, risulta difficilmente controllabile una situazione dove tutta l'informazione viene raccolta in più fogli elettronici. Nella situazione attuale, una possibile dissonanza fra le informazioni archiviate e la realtà può nascere da una semplice sostituzione di una lampada, dove l'ufficio Manutenzioni si occupa della sostituzione mentre l'ufficio Lavori Pubblici detiene il foglio elettronico. Nel caso in cui la nuova lampada montata sia diversa da quella installata originariamente e se, contemporaneamente, il primo ufficio non lo comunica al secondo, il difetto della procedura attuale si rifletterebbe immediatamente in termini di correttezza dei dati.
Inoltre, a tutto ciò vanno aggiunti quelli che sono degli obiettivi specifici che devono soddisfare gli utilizzatori finali. In particolare, essi sono:

1. poter visionare la potenza realmente consumata in ciascun quadro, in modo da valutare se, in base al loro posizionamento e mantenendo inalterata la disposizione dei punti luce, è possibile fare una sorta di accorpamento che permetta di risparmiare energia;
2. sapere quali punti luce montano lampade ad incandescenza, in modo tale da poter stimare, con buona precisione, l'entità del vantaggio economico derivante da una possibile sostituzione con lampade a basso consumo;
3. ricercare quegli impianti di illuminazione che vanno adeguati alla nuova normativa (Legge Regionale n.17 del 7/8/2009), di prossima applicazione, dove si stabiliscono i criteri di “riduzione dell'inquinamento luminoso e ottico, nonché la riduzione dei consumi energetici da esso derivanti”.

3.2 REQUISITI

La progettazione dell'intervento sul sistema in uso deve tenere in considerazione varie esigenze.

Essa deve prevedere un archivio che memorizzi tutti i dati riferiti alla rete di illuminazione pubblica. In particolare, il modello dati che verrà creato dovrà tener conto della necessità di censire tutti i *quadri elettrici* presenti nella rete e, nello specifico, memorizzarne:

- un numero identificativo;
- la via in cui si trovano;
- la tipologia (esempio: trifase);
- il numero di linee elettriche che vi partono;
- il tipo d'impianto (esempio:derivazione);
- la tensione;
- la potenza contrattuale;
- i *dati riguardanti lo storico fornitura dell'energia elettrica*.

In riferimento all'ultimo punto, va specificato che, per coloro che operano in quest'ambito, è di interesse mantenere una semplice cronologia, senza specifici riferimenti temporali, che permetta di risalire solamente al fornitore dell'energia e al numero di contratto.

Ai quadri elettrici appena descritti sono collegati dei *punti luce*, di cui interessa:

- assegnare un identificativo numerico;
- la via dove sono situati;
- la tipologia (cioè se è un palo o qualcos'altro);
- se è presente o meno un pozzetto;
- l'altezza;
- lo sbraccio;
- il tipo di linea con cui vengono alimentati (cavidotto o linea aerea);
- la lunghezza della linea (cioè la distanza tra il punto luce e il quadro a cui è collegato);
- il numero dei cavi della linea e la loro sezione;
- le *lampade* montate, di cui interessa la marca, la potenza, la tipologia (es: ad

incandescenza) e saperne l’uso che ne viene fatto, cioè se sono poste alla sommità dei pali o se vengono montate all’interno di fari;
- il quadro a cui è collegato;
- un attributo su cui poter annotare delle osservazioni.

Per quanto riguarda le lampade montate su un punto luce, va osservato che, in linea di massima, anche quando ne sono presenti più di una sullo stesso punto, sono identiche fra loro.
La struttura dell’archivio che verrà implementato dovrà permettere di risalire alle vie che vengono alimentate e alla reale potenza utilizzata in un determinato quadro elettrico.

Inoltre, sul database che verrà progettato e realizzato, in base a queste indicazioni, dovranno essere costruite delle applicazioni PHP che forniscano un’interfaccia semplice ed intuitiva, tale da permettere, a tutti i soggetti che avranno bisogno di accedervi, la possibilità di modificarne e consultarne il contenuto. La creazione di questo insieme di applicazioni va corredato da un sistema di autenticazione che dia la possibilità di personalizzare, in base al profilo a cui appartiene colui che effettua l’accesso, le funzionalità messe a disposizione.

Realizzato il tutto, deve essere predisposto un sistema cartografico dove siano rappresentati i vari elementi che costituiscono la rete di illuminazione pubblica. Tale sistema si interfaccerà con il DB creato, al fine di poter visualizzare, direttamente sulle mappe, le informazioni su un certo elemento.

4. SCELTE TECNOLOGICHE

4.1 POSTGRESQL – POSTGIS

PostgreSQL è un ORDBMS (*Object-Relational DataBase Management System*, cioè un database relazionale ad oggetti), rilasciato con licenza libera.

PostgreSQL, pur presentando caratteristiche classiche (come l'utilizzo del linguaggio SQL e i meccanismi propri dell'integrità referenziale), presenta una peculiarità di particolare importanza: la *programmabilità*.

Questa consiste nella possibilità di passare dal DB, in cui le informazioni sono definite usando i tipi di dato standard del SQL, ad una filosofia dove il dato assume una forma più strutturata, come avviene in molti linguaggi di alto livello. Una soluzione proposta inizialmente per risolvere questo tipo di problema consisteva in una sorta di mappatura che, con il tempo, è risultata molto inefficiente sotto l'aspetto del costo computazionale. Questo problema è stato affrontato e risolto in PostgreSQL dando la possibilità all'utente di definire un nuovo tipo di dato come insieme di combinazioni dei tipi presenti nel linguaggio SQL di partenza. Tutto ciò ha fatto si che il dato strutturato fosse di immediata comprensione per il DB. Le potenzialità di questo modo di organizzare i dati possono essere comprese tramite un semplice esempio: se si definisce l'informazione indirizzo come l'insieme di via, numero civico, codice di avviamento postale, città e provincia, tale struttura può essere definita in ciascuna tabella richiamando una singola riga di codice. Mentre all'inizio, per poter utilizzare questo tipo di funzionalità, era necessario scrivere delle estensioni in linguaggio C, ora è possibile definire un dato strutturato mediante l'esecuzione del comando CREATE DOMAIN.

I vantaggi derivanti dalla programmabilità possono consistere anche nella possibilità di definire delle funzioni in SQL, richiamabili in qualsiasi punto da altri codici. Va comunque specificato come questo tipo di linguaggio sia, per sua natura, poco adatto a soddisfare certi aspetti, visto che strumenti come i cicli non sono previsti. Per questo in PostgreSQL, come in molti altri DBMS, sono state apportate delle modifiche alla forma canonica del SQL, per arricchirlo con delle peculiarità presenti nella maggior parte dei linguaggi di programmazione. A questo punto è facile immaginare come diventi implementabile un meccanismo in cui vengono definite alcune procedure standard sul lato server, che possono essere richiamate sul lato

client da chiunque ne abbia bisogno.
Gli altri buoni motivi, oltre a quello appena illustrato, che hanno portato l'ente a scegliere questo prodotto vengono elencati di seguito:

- è multipiattaforma, in quanto gira su tutti i più importanti sistemi operativi;
- può essere utilizzato con gran parte dei linguaggi di programmazione;
- ha diversi strumenti di gestione, come phpPgAdmin;
- ha un buon servizio di supporto;
- è un DBMS che può gestire basi di dati di grandezza illimitata;
- è gratuito;
- è Open Source.

Un concreto utilizzo della programmabilità offerta da PostgreSQL è stato fatto nella realizzazione di una delle sue più importanti estensioni: *PostGIS*.
Tale componente da la possibilità di gestire gli *oggetti cartografici*, trattandone tutti quegli aspetti caratteristici che diversamente non potrebbero essere trattati in un normale DBMS. In questo modo vengono memorizzati oggetti come punti, linee, poligoni e tanti altri elementi geometrici/geografici.
La tipologia di dati attualmente utilizzati si classificano in 2 tipi:

- GEOMETRICI, dove l'informazione memorizzata è basata su un insieme di coordinate del piano, quindi in un sistema a 2 dimensioni;
- GEOGRAFICI, che invece lavora su un sistema di coordinate di tipo geodetico, basato sul principio che considera che la Terra non ha una forma planare bensì sferica.

Va specificato che l'unico standard utilizzato per i dati GEOGRAFICI in questo ambiente è WGS84 che, oltre ad essere adoperato anche in ambito aereonautico, ha la grande qualità di permettere il passaggio da un tipo di dato all'altro.

Questa distinzione è di fondamentale importanza dato che, per esempio, nel misurare la distanza che intercorre tra 2 punti distinti della superficie terrestre il risultato può variare più o meno significativamente. Infatti, nel caso in cui i punti siano posizionati in zone diametralmente opposte, i dati geografici ci danno modo di misurare direttamente l'arco che collega le 2 località e non, come avverrebbe in ambito geometrico, il segmento che li unisce dopo averli proiettati sul piano. Lo scenario

appena esposto ci da il modo di apprezzare come l'ambito geometrico, pur disponendo di un numero maggiore di funzioni, possa introdurre nelle misurazioni un effetto distorcente abbastanza rilevante.
D'altro canto, l'ambito geometrico da la possibilità di eseguire certi calcoli, come l'area e la distanza, con un costo computazionale minore. Tutto ciò da l'idea del perché il numero di operazioni a cui possono essere soggetti i dati geometrici sia di gran lunga maggiore rispetto a quelle sui dati geografici.
Sulla base di questi aspetti, risulta utile capire quando conviene mappare i dati in forma geometrica e quando in forma geografica. In tal senso:

- se si fa riferimento ad una piccola area è meglio scegliere la forma geometrica, visto che le misure sono sufficientemente fedeli e le operazioni sui dati sono meno onerose;
- al contrario, se si tratta di un'area continentale è preferibile orientarsi sulla forma geografica, che permette di evitare problemi derivanti da proiezioni sul piano di aree abbastanza estese;
- se le conoscenze in ambito geometrico e le esigenze di operare sui dati sono limitate, è consigliabile basarsi sulla rappresentazione geografica.

I tipi di informazione appena esposti trovano collocazione all'interno del sistema in apposite tabelle, dove gli oggetti GIS inseriti vengono memorizzati in una forma che, anche se può variare da caso a caso, è in linea di massima la seguente:

Id	Nome	Geom

Mentre i primi 2 sono dei normali campi in cui sono contenuti, rispettivamente, un identificativo e un nome dell'oggetto, l'attributo "Geom" indica quella struttura articolata che è designata a ricevere e contenere i dati cartografici riferiti ai vari componenti che vengono mappati.

Grazie al contesto appena esposto, una volta raccolti ed inseriti i dati, è possibile avvalersi di funzioni interessanti, mediante la stesura, sullo stesso principio dei normali Database, di query spaziali. Queste permettono di rispondere a quesiti come la distanza fra 2 località o l'individuazione, fissato un punto di riferimento, degli oggetti situati all'interno di un certo raggio.

Nel modellare i vari tipi di oggetti, PostGIS fa riferimento, in prima istanza, a quelli che sono 2 standard: WKT (Well-Known Text) e WKB (Well-Known Binary). Ambedue si basano sul concetto di memorizzare il tipo di oggetto e la sua posizione, che assume significato nell'ambito di un ben determinato sistema di coordinate, che va specificato tramite un identificatore di sistema di riferimento spaziale (SRID - Spatial Referencing system IDentifier).
Visto il limite spaziale posto da questo tipo di approccio, si è deciso di estendere questi standard (chiamati EWKT e EWKB), in modo tale da riuscire a dare una rappresentazione anche a quegli oggetti che si presentano in 3 o, addirittura, in 4 dimensioni.
Adesso è arrivato il momento di vedere, un po' più nello specifico, la struttura che implementa tutte le funzionalità e utilizza gli standard sopra menzionati.

PostGIS segue quelli che sono i dettami sanciti dagli OpenGIS Standard, stabiliti dal consorzio OpenGIS (OGC). Tali principi affermano la necessità della presenza di 2 tabelle: *SPATIAL_REF_SYS* e *GEOMETRY_COLUMNS*.
La prima è una tabella che contiene più di 3000 sistemi di riferimento spaziale, assieme a tutti i particolari necessari per proiettare le coordinate di partenza in uno degli altri sistemi conosciuti. Inoltre, va specificato come tale tabella non costituisca una sorta di insieme chiuso, ma permetta di essere arricchita con nuove informazioni riguardanti sistemi di coordinate non ancora presenti. Questo grado di personalizzazione risulta essere di vitale importanza, visto che la realtà che ci si trova a modellare vede l'adozione di specifici modelli di riferimento per ogni area geografica. La struttura scelta per la tabella in questione è:

```
CREATE TABLE spatial_ref_sys (
      srid       INTEGER NOT NULL PRIMARY KEY,
      auth_name  VARCHAR(256),
      auth_srid  INTEGER,
      srtext     VARCHAR(2048),
      proj4text  VARCHAR(2048)
);
```

- `SRID`: valore intero con cui il sistema di riferimento spaziale è identificato all'interno della tabella;
- `AUTH_NAME`: nome dell'organizzazione di cui fa parte il sistema di riferimento;
- `AUTH_SRID`: identificativo associato al sistema dall'organizzazione citata in

AUTH_NAME;

- SRTEXT: indicazione sulla rappresentazione WKT adoperata dal sistema di riferimento spaziale;
- PROJ4TEXT: contiene i riferimenti ad una libreria, chiamata Proj4, che fornisce i dettagli per effettuare la trasformazione delle coordinate.

La tabella GEOMETRY_COLUMNS, invece, presenta quest'altra strutturazione:

```
CREATE TABLE geometry_columns (
      f_table_catalog         VARCHAR(256) NOT NULL,
      f_table_schema          VARCHAR(256) NOT NULL,
      f_table_nam             VARCHAR(256) NOT NULL,
      f_geometry_column       VARCHAR(256) NOT NULL,
      coord_dimension         INTEGER NOT NULL,
      srid                    INTEGER NOT NULL,
      type                    VARCHAR(30) NOT NULL
);
```

- F_TABLE_CATALOG, F_TABLE_SCHEMA, F_TABLE_NAME: l'insieme di questi 3 campi costituiscono il nome completo della tabella, cioè il percorso in cui si trova. In PostgreSQL il primo campo rimane bianco, dato che il concetto di catalogo non è presente;
- F_GEOMETRY_COLUMN: il nome della colonna “geometrica” della tabella;
- COORD_DIMENSION: la dimensione spaziale (2, 3 o 4);
- SRID: id assunto dal sistema di riferimento in SPATIAL_REF_SYS;
- TYPE: il tipo di oggetto (POINT, POLYGON, ecc.).

Fatta questa premessa, il procedimento seguito per definire una tabella spaziale consiste nel crearla, invocando successivamente su di essa la funzione `AddGeometryColumn( <table_name>, <column_name>, <srid>, <type>, <dimension>)`. Per esempio:

```
CREATE TABLE parks (
      park_id    INTEGER,
      park_name  VARCHAR,
      park_date  DATE,
      park_type  VARCHAR
);
SELECT AddGeometryColumn('parks', 'park_geom', 128, 'MULTIPOLYGON',
2 );
```

In questo caso, si vede come prima venga creata la tabella `parks` con i suoi campi e, solo successivamente, come venga aggiunta la colonna `park_geom`, che sarà definita per contenere informazioni su MULTYPOLIGON in 2 dimensioni, nel sistema di riferimento con SRID 128.

L'idea che sta dietro alla definizione di questa funzione è quella di una procedura automatica che si occupi, contemporaneamente, della creazione della colonna nella specifica tabella e di inserire le informazioni riguardanti tale colonna in `GEOMETRY_COLUMNS`.
Questa importante funzionalità ha però il limite di non poter essere applicata a tutti i costrutti, come nel caso delle viste. In questa situazione l'unica via percorribile è quella di andare ad inserire “manualmente” i dati nella tabella `GEOMETRY_COLUMNS`. In tal senso, vediamo un esempio per capire il procedimento che va seguito:

```
--Data la vista
CREATE VIEW  public.vwmytablemercator AS
SELECT gid, ST_Transform(the_geom,3395) As the_geom, f_name
FROM public.mytable;

--procedimento per registare la vista in GEOMETRY-COLUMNS
INSERT   INTO   geometry_columns(f_table_catalog,   f_table_schema,
f_table_name, f_geometry_column, coord_dimension, srid, "type")
SELECT    '',    'public',    'vwmytablemercator',    'the_geom',
ST_CoordDim(the_geom), ST_SRID(the_geom), GeometryType(the_geom)
FROM public.vwmytablemercator LIMIT 1;
```

Nella seconda parte del codice SQL si definisce l'inserimento in `GEOMETRY_COLUMNS` del risultato di un interrogazione che, oltre ad inserire dei parametri già definiti, si avvale delle funzioni `ST_CoordDim(the_geom)`, `ST_SRID(the_geom)` e `GeometryType(the_geom)`, che servono rispettivamente per ottenere la dimensione delle coordinate, SRID e il tipo di oggetto (es: POINT).

4.2 MAPSERVER

Mapserver è un ambiente Open Source, che funziona su tutti i sistemi operativi più conosciuti, sviluppato con lo scopo di rappresentare graficamente dei dati geospaziali.
Può essere utilizzato come supporto per applicazioni WebGIS, che saranno trattate in seguito, e come strumento per fornire servizi conformi agli standard definiti dall'OGC (Open Geospatial Consortium). Fra questi, i più importanti sono:

- *Web Map Service* (WMS) che, partendo da informazioni di carattere geografico e seguendo procedure dinamiche, restituisce mappe sottoforma di immagini digitali pronte per essere visualizzate su un browser;
- l'interfaccia *Web Feature Service* (WFS) usata dal client per richiedere oggetti cartografici via Web.

L'interazione fra queste due tecniche permette di definire degli oggetti geografici e, facendo una richiesta tramite l'interfaccia WFS ad un WMS (o un portale geografico come Google Maps), sovrapporvi le corrispettive mappe.

Mapserver è un programma CGI (Common Gateway Interface – standard che definisce il modo con cui un server web può delegare ad apposite applicazioni la creazione di pagine dinamiche) che resta inattivo nel server web in attesa di una richiesta. Quando ciò avviene, usa le informazioni che gli vengono passate, assieme a quelle che ha già, per restituire l'immagine della mappa richiesta.

I dati che sono già in possesso dell'ambiente vengono memorizzati in appositi *mapfile*, che sono dei file con estensione *.map* . Questi sono file di testo strutturato che definiscono tutta la configurazione e, nello specifico:

- l'area della mappa;
- dove vanno cercati certi dati;
- dove posizionare l'immagine in output;
- i layer, con la sorgente dei dati, le proiezioni e il tipo di simbologia.

Da questo elenco si può capire come questi file siano essenziali per l’acceso ai dati e lo styling dei vari layer previsti.

Per capire meglio ciò di cui si sta parlando, può essere utile vedere un esempio dove si fissano i parametri necessari per la “costruzione” di un layer, basandosi sui rispettivi dati archiviati in PostGIS.

```
LAYER
  CONNECTIONTYPE postgis
  NAME "widehighways"
  # connessione al db spaziale remoto
  CONNECTION "user=dbuser dbname=gisdatabase host=bigserver"
  PROCESSING "CLOSE_CONNECTION=DEFER"
  # si fissa qual è la tabella e la colonna da cui pescare i dati
  DATA "geom from roads using srid=4326 using unique gid"
  STATUS ON
  TYPE LINE
  # visualizzare solo le autostrade (strade con almeno 4 corsie)
  FILTER "type = 'highway' and numlanes >= 4"
  CLASS
    # Aumentare la luminosità delle autostrade più grandi e fissare a 2
      # pixel la loro larghezza
    EXPRESSION ([numlanes] >= 6)
    STYLE
      COLOR 255 22 22
      WIDTH 2
    END
  END
  CLASS
    # il resto più scure e con 1 pixel di larghezza
    EXPRESSION ([numlanes] < 6)
    STYLE
      COLOR 205 92 82
    END
  END
END
```

Specifichiamo il significato di alcuni comandi:

- `CONNECTIONTYPE`: tipo di DB a cui ci si connette;
- `CONNECTION`: tutti i dati riguardanti la connessione;
- `DATA`: si specificano tabella e colonna in cui sono presenti i dati spaziali;
- `FILTER`: è una comando con cui si filtrano i dati che devono far parte del layer, sulla base di alcune caratteristiche che devono essere rispettare. Di fatto, è abbastanza analogo a ciò che avviene quando si inseriscono delle discriminanti nella clausola WHERE.

Da questo esempio è possibile rendersi conto della potenza di un strumento che permette di definire, una volta che l'utente abbia richiesto di vedere le autostrade, dove risiedono i dati e come essi devono essere presentati a chi ne ha fatto domanda.

MapServer può essere ulteriormente arricchito grazie all'utilizzo di specifici *MapScript*. Questi forniscono delle interfacce per la costruzione di applicazioni, siano esse finalizzate o meno al web. Ciò significa che un MapScript è un modulo che può essere aggiunto a MapServer per fornire ulteriori opportunità in ambito di linguaggi di scripting.

Un'altra parte essenziale è quella riferita ai *dati geografici*. Queste informazioni possono essere rappresentate secondo la *grafica raster* o quella *vettoriale*, che sono attualmente quelle più usate per definire le immagini in ambito informatico.
La prima di queste è una tecnica che vede l'immagine come una griglia ortogonale di punti, chiamata proprio raster. In questa logica, ad ogni punto di questa sorta di scacchiera, chiamato pixel, viene associato un singolo colore, che è il risultato della gradazione di tre "colori primari": rosso, verde e blu (tecnica RGB). Queste immagini si basano su 2 concetti:

- la risoluzione, che specifica il numero di pixel che compongono un'immagine;
- la profondità di colore, che dipende essenzialmente dal grado di accuratezza con cui viene memorizzata l'informazione, cioè dai bit utilizzati per descrivere ciascun pixel.

La grafica raster risulta poco vantaggiosa se si svolgono certe operazioni. Questo si riscontra soprattutto quando si fanno degli zoom che, per la natura della tecnica in questione, danno come risultato delle immagini che possono risultare significativamente sgranate. Nonostante esistano programmi che si occupano di aggiungere dei pixel per mitigare questa problematica, tale operazione è comunque sconsigliata perché porta il software a fare delle supposizioni, senza che l'utente possa sapere nulla in merito alla loro correttezza. I vantaggi, al contrario, si riscontrano nelle rappresentazioni della realtà (ortofoto) e nella modifica di alcuni parametri, come contrasto e luminosità. Nello specifico, in ambito GIS, il termine raster viene usato per indicare la tipologia di dato impiegata nella rappresentazione cartografica digitale. Con i dati raster il territorio viene riprodotto attraverso una matrice di pixel di forma quadrata o rettangolare. A ciascun pixel è associato un

attributo che definisce le caratteristiche dell'elemento rappresentato. Questo permette, ad esempio, di memorizzare l'elevazione di ciascun pixel, associando ad ogni punto il valore della quota sul livello del mare. Per ciò che è stato detto in precedenza, la dimensione dei pixel è inversamente proporzionale alla precisione della carta. I dati raster possono essere acquisiti in un sistema GIS mediante l'uso di apparecchiature a lettura ottica, come ad esempio scanner, o attraverso l'elaborazione di dati, raster o vettoriali, già acquisiti via satellite.

Nella *grafica vettoriale*, invece, le unità elementari con cui sono rappresentate le immagini non sono i pixel ma una serie di primitive geometriche, come punti, linee, curve e poligoni. Questo metodo da la possibilità di archiviare l'informazione in base a delle equazioni matematiche, apportando a questo modello dei considerevoli vantaggi. Uno su tutti si ha quando si prova a rappresentare un poligono: esso viene definito come una serie di punti (i vertici), identificati da delle coordinate e collegati da alcune linee. Ciò diventa di fondamentale importanza quando il poligono viene visualizzato su dispositivi con risoluzione differente. Infatti, con questa tecnica l'immagine avrà sempre lo stesso grado di definizione, visto che è totalmente sganciata dal concetto di pixel. Al contrario, come abbiamo visto in precedenza, la stessa cosa non può essere detta per la grafica raster, dove le varie linee verrebbero rappresentate come una serie di pixel, comportando il rischio di segmenti non ben definiti se l'immagine viene visualizzata su dispositivi ad alta risoluzione. Un secondo vantaggio, non meno importante, si ha nella quantità di memoria richiesta per memorizzare i dati. Infatti, sempre rifacendosi ad un esempio geometrico, come quello di una linea, è sufficiente che vengano riportate coordinate d'inizio e fine, con l'equazione matematica che la definisce. Parallelamente, se si prova ad archiviare la medesima informazione nel formato raster, sarebbe richiesta una porzione di memoria dedicata per ogni pixel che contribuisce a formare tale linea. Le proprietà appena esposte sono importanti a tal punto che alcuni dati cartografici, noti per la loro complessità ad essere manipolati, vengono gestiti con questa logica. Basti pensare a quando si rendono necessarie operazioni di cambio scala e a quanto sarebbe complicato gestire il tutto in ambito raster. Ma, per la serie “non è tutto oro quello che luccica”, questa tecnica comporta anche dei seri limiti al suo utilizzo. Infatti, la composizione di questo tipo di immagini risulta essere molto meno intuitiva di quella raster, visto che è necessario avere delle competenze geometriche approfondite per sfruttare a pieno i vantaggi appena elencati. Un altro elemento detrattore nei confronti di questo modello è l'unità di calcolo di cui si deve disporre.

La causa risiede nel fatto che, per descrivere un'immagine nei suoi aspetti anche più particolari, ci si può scontrare con delle equazioni matematiche abbastanza complesse che, se da una parte consentono un risparmio in termini di memoria, dall'altro comportano un costo computazionale significativo nell’essere elaborate.

Per interloquire con tutte le componenti della logica adottata da Mapserver, viste fino a qui, viene previsto un file eseguibile che si occupa di restituire, in base alle richieste fatte, immagini e dati. Va specificato che la directory in cui risiede questo file non è la Web root, poiché l'utente che fa la richiesta dove poter accedere a tale directory mentre la Web root, per questioni di sicurezza, non può farsi carico di questo tipo di necessità.

Infine, si ha un HTTP server che fornisce la risposta, sottoforma di pagina HTML, agli utenti che l’hanno richiesta. Le piattaforme che possono ricoprire questo ruolo, che vanno installate sulla medesima macchina in cui risiede MapServer, sono Apache e Microsoft Internet Information Server (IIS).

Di seguito uno schema che riassume le relazioni fra le varie componenti di MapServer:

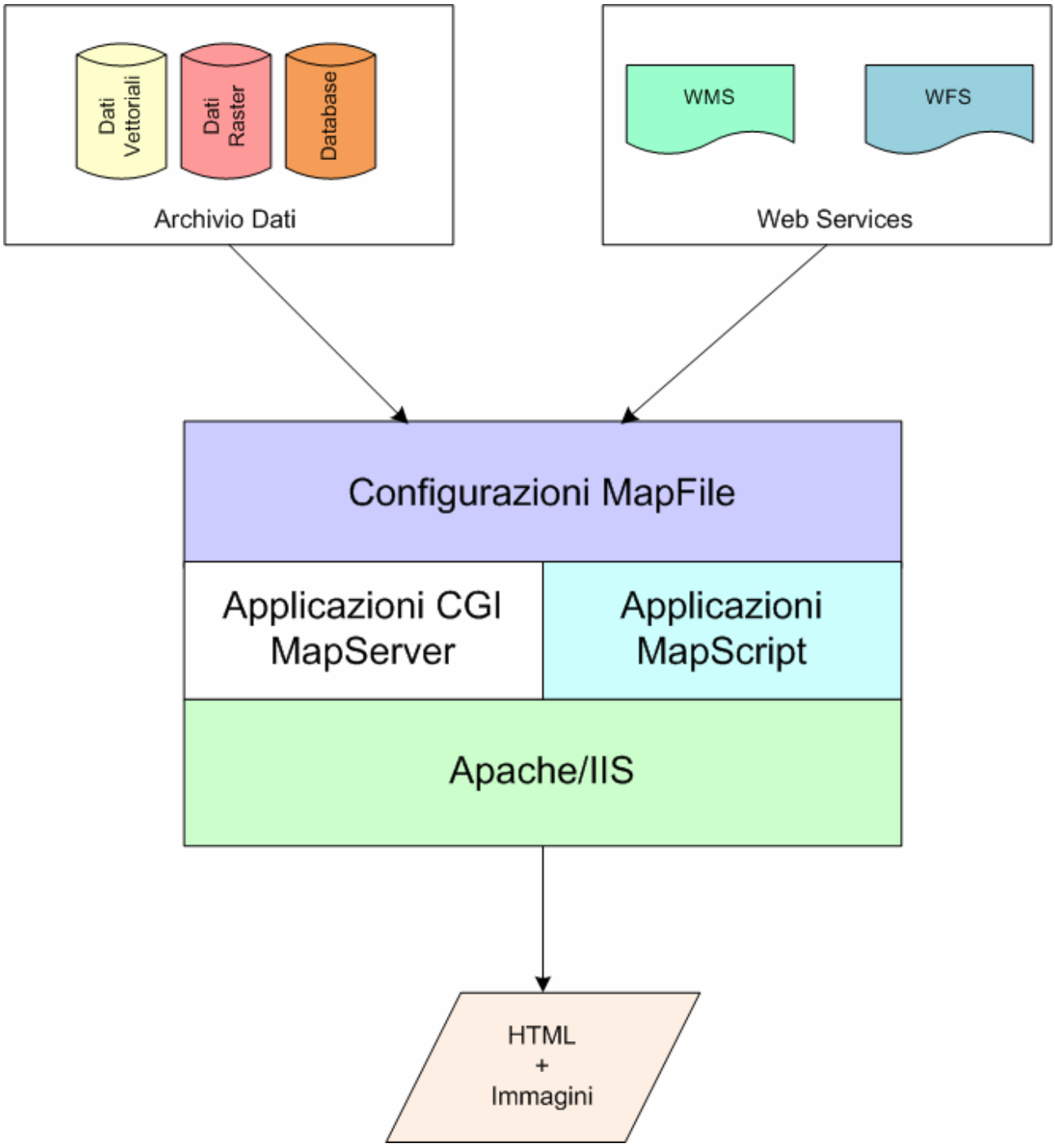


Figura 4.1 – Struttura MapServer

4.3 INFORMCITY

InformCity è un Sistema Informativo territoriale via Web (WebGIS), sviluppato sulla base di tecnologie OpenSource (MapServer, PostgreSQL e PostGIS), che consente di affrontare in modo integrato aspetti ambientali ed urbanistici per la gestione del territorio.

Con il termine *WebGIS* si indicano i Sistemi Informativi Territoriali pubblicati sul web che forniscono le estensioni per gli applicativi operanti sulla cartografia. Le potenzialità principali di queste tipologie di strumenti risiedono nella possibilità di fornire all'esterno mappe arricchite da informazioni di vario genere. Ciò risulta essere molto importante per una pubblica amministrazione che, tramite questi prodotti, ha la possibilità di fornire alla cittadinanza servizi che possono essere utilizzati avvalendosi di un semplice browser web, talvolta corredato da alcuni plug-in. I vantaggi sono ambivalenti: per i cittadini, che possono acquisire le informazioni stando semplicemente davanti al proprio pc, e per gli uffici pubblici, che si trovano sgravati del compito di rispondere a certi quesiti. Gli ambiti di applicazione che possono dare risultati interessanti sono elencati di seguito:

- **gestione del territorio**, mediante l'incrocio di dati cartografici, provenienti da settori differenti (esempio: piano regolatore sovrapposto alle foto aeree);
- **pianificazione**, con cui è possibile fare un raffronto fra la situazione attuale e una possibile situazione futura, basata su una ben precisa ipotesi di intervento;
- **classificazione territoriale a fini fiscali**, dove si vanno ad individuare le varie zone catalogate in base all'ICI dovuto o agli eventuali benefici spettanti;
- **informazione agli utenti**, tramite un sistema di accesso controllato ai dati, mediante il quale è possibile fare una distinzione fra i dati di interesse per i normali cittadini e quelli necessari allo svolgimento di specifiche professioni (geometri, architetti, ecc.).

Quest'ultima distinzione degli utenti in categorie è resa possibile grazie al fatto che l'accesso al sistema è consentito solo tramite l'uso di credenziali. È proprio quest'approccio che permette di definire, all'amministratore del sistema, varie categorie di utenti, assegnando a ciascuna l'utilizzo di funzioni specifiche.

4.4 MECCANISMO INTERAZIONE DEI PACCHETTI

Dopo aver elencato le proprietà dei vari pacchetti che costituiscono la soluzione al problema, è arrivato il momento di dare un'idea su come questi interagiscono fra loro.

Alla base della struttura si colloca *PostgreSQL/PostGIS*, dove sono contenute le informazioni sui vari elementi (quadri elettrici, punti luce). In tal senso, tenendo presente la struttura adottata per memorizzare gli oggetti in ambito GIS, il più importante valore è quello contenuto nel campo "*Geom*", che è una stringa alfanumerica che ha il compito di indicare la posizione precisa in cui si trovano i vari elementi della rete di illuminazione pubblica. Questi dati vengono memorizzati importando uno *shape* in PostGIS tramite l'uso di *MapStorer*, che è un applicativo del pacchetto *Informcity*. Il risultato di questa operazione è una tabella contenente una tupla (con i campi obbligatori *gid* e *the_geom*, più quelli aggiunti a seconda delle necessità) per ogni elemento presente nello shape.

Successivamente, sempre mediante l'utilizzo di MapStorer, si procede alla vestizione grafica delle informazioni contenute nei vari shape importati. Tale fase genera un particolare codice, che verrà inserito in un *Mapfile* (con estensione *.map*) destinato ad essere collocato all'interno della struttura di *MapServer*, illustrata in precedenza. Infine, prima di poter rendere fruibile il servizio, si definiscono, a livello di *Informcity*, le regole di accesso ai dati, stabilendo gli utenti che vi possono entrare e, per ciascuno di essi, i layer che sono autorizzati a visionare e le operazioni che possono fare su di essi.

A questo punto, quando arriva una richiesta al sistema, Informcity, dopo aver gestito la parte riguardante l'autenticazione, chiede a MapServer, via api, di costruire la parte cartografica della risposta che va restituita all'utente. Quest'ultimo comincia a formulare la risposta raccogliendo una serie di informazioni:

- le *ortofoto*, che possono provenire da fonti interne o da terzi (sfruttando le tecniche *WMS*-*WFS*);
- la *posizione di ciascun elemento* contenuto all'interno dei layer che si vogliono visualizzare. Ciò si recupera andando a pescare, all'interno di PostGIS, le stringhe alfanumeriche contenute nel campo *the_geom*;
- la *veste grafica* con cui i vari elementi devono apparire. Per sapere questo, MapServer va a cercarsi proprio quei *Mapfile* riferiti ai layer che sono stati richiesti da Informcity.

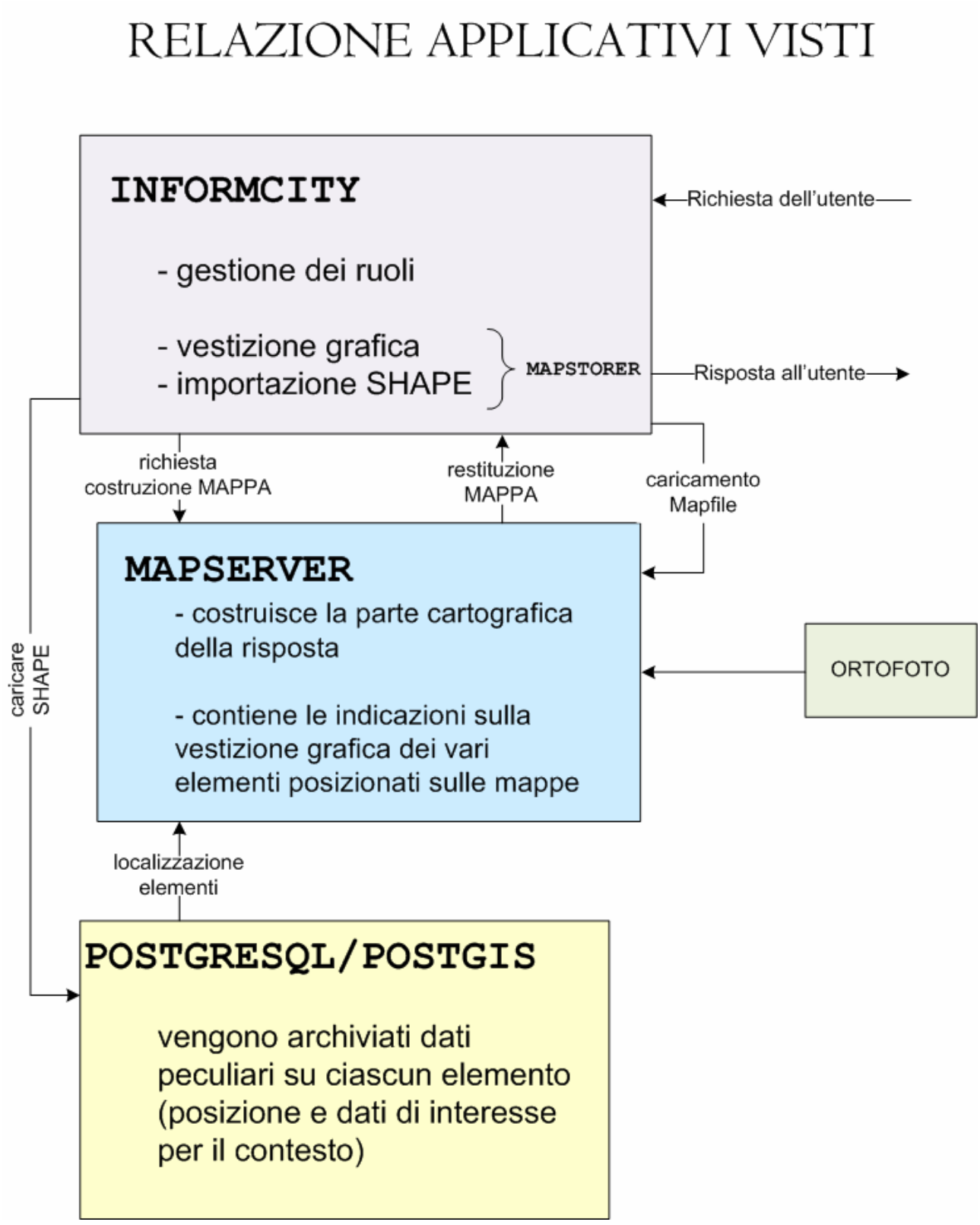


Figura 4.2 – Interazione fra i vari componenti del sistema

4.5 SCRIPTCASE

Questo applicativo è un ambiente di sviluppo usato per generare programmi in codice PHP. La peculiarità di questo prodotto sta nel fatto che è implementato in modo tale da funzionare direttamente su browser, permettendo una più facile condivisione dello sviluppo di progetti fra i vari componenti di un team. Infatti, installando questo software sul server di una rete locale è possibile, per tutti coloro che ne abbiano la necessità, collegarsi alla intranet aziendale e usufruire di questo strumento.

Questo pacchetto offre il supporto per la connessione con tutti i database più conosciuti (Oracle, DB2, MS SQLServer, MySQL, PostgreSQL, Sybase, MS Access, ecc.) e le applicazioni che genera sono compatibili sia con piattaforme Windows che con quelle Linux. Questo vantaggio nasce dalla struttura stessa dell'ambiente, dove il codice generato è totalmente indipendente dal Sistema Operativo, dato che il linguaggio PHP, essendo libero, può essere utilizzato su qualsiasi piattaforma.

I vari applicativi generabili usando ScriptCase si possono classificare in categorie, fra cui le più importanti sono:

- *grid*, dove i dati vengono visualizzati in sola lettura ed è possibile fare ricerche, ordinamenti, esportazione di dati in vario formato e generare reportistica in formato PDF;
- *menu*, usati per fornire una grafica sufficientemente intuitiva, dove l'utente finale abbia l'opportunità di selezionare le varie applicazioni semplicemente cliccando sulla voce presente nei menu a tendina;

- *tree menu*, simile al precedente, con la sostanziale differenza che le varie voci che lo costituiscono sono visualizzate in modo analogo a ciò che avviene con la rappresentazione della struttura delle cartelle;
- *form*, con cui vengono inseriti e/o modificati i dati contenuti nelle tabelle;
- *tab*, usate per avere modo di richiamare più applicazioni all'interno della medesima schermata, attraverso la suddivisione in schede;
- *control*, programma usato per definire le form di autenticazione, assieme a tutti quei codici di controllo che regolano tale meccanismo;
- *PDF report*, usati per impostare la costruzione di file PDF, contenenti informazioni di reportistica sui valori assunti da alcuni campi, specificati in fase di definizione.

Per le caratteristiche appena esposte, ScriptCase è lo strumento ideale per implementare l’interfaccia utile a rendere il database consultabile e modificabile da tutti, in modo facile ed intuitivo.

5. PROGETTAZIONE

5.1 SCHEMA CONCETTUALE

Adesso, dopo una presentazione generale sui sistemi già in uso e su quelli che verranno adottati prossimamente, è arrivato il momento di progettare concettualmente, sulla base degli obiettivi e dei requisiti già esposti, il database necessario a gestire le informazioni nel caso preso in esame.

Innanzitutto va prevista la presenza dell'entità *Punto Luce*, che rappresenta i vari nodi della rete utilizzati per illuminare vie, piazze e viali pedonali. Questi sono caratterizzati da un codice identificativo, il tipo (es: palo), la presenza o meno di un pozzetto, l'altezza a cui è situato, lo sbraccio e eventuali annotazioni.

Tale entità deve collegarsi a *Quadro,* in cui si indicano i quadri elettrici deputati ad alimentare i vari punti luce della rete di illuminazione pubblica e di cui interessa avere un identificativo numerico, il tipo (es: trifase), il tipo di impianto (es: derivazione), il numero di linee che vi escono, tensione e potenza disponibile. Le 2 entità appena descritte vengono collegate con l'associazione *Allaccio*, in cui vengono specificate il tipo di linea con cui sono collegati (aerea o interrata) e il numero di cavi di tale linea, con la loro sezione e la loro lunghezza (cioè la misurazione del tratto che collega il punto luce al suo quadro elettrico).

Nel contesto descritto fino a qui si vuole conoscere su quale strada sono ubicati i vari punti luce e quadri elettrici. Questo è reso possibile con la presenza delle associazioni *Via Quadro* e *Via Punto Luce*, con cui, rispettivamente, *Quadro* e *Punto Luce* vengono collegati a *Stradario*. Quest'ultimo è un'entità già presente nel sistema informativo in uso ed ha lo scopo di censire tutte le vie presenti nel comune di Feltre, mediante un codice numerico e il nome della via.

Si prevede anche la presenza di *Lampada* per poter rappresentare l'intero inventario di tutte le lampade dislocate all'interno della rete comunale. In questa situazione, ciò che va mappato è un identificativo e l'uso che viene fatto di tale articolo. In tal senso, è importante ricordare che una lampada può essere posta alla sommità di un palo o all'interno di un faro. Tale entità, come è logico che sia, viene collegata a *Punto Luce* mediante l'associazione *Composizione Punto Luce,* che si occupa di registrare quali sono le lampade in uso su un determinato punto luce. In questa situazione viene anche prevista la presenza di *Modello Lampada* con lo scopo di classificare i vari

esemplari utilizzati, in base a caratteristiche come la marca, la potenza e la tipologia (es: ad incandescenza).

Va inoltre creato un sistema che consenta di tenere traccia di tutti i contratti di fornitura che si sono susseguiti per i vari quadri elettrici. Ciò viene permesso con l'introduzione di *Storico Forniture*, che ha gli attributi codice, fornitore, numero contratto e l'indicazione della data iniziale ed, eventualmente, finale del contratto in questione. Collegando questa nuova entità a Quadro, tramite *Cronologia Forniture*, sarà possibile ottemperare all'esigenza di sapere per ogni quadro elettrico, oltre al contratto di fornitura attuale, anche tutti quelli che l'hanno preceduto.

Quello che è stato previsto fino a qui non contempla ancora la necessità di disporre della localizzazione precisa su mappa dei quadri elettrici e dei punti luce.

Per implementare tutto questo si procede con l'introduzione di 2 entità, *Posizione Quadro* e *Posizione Punto Luce*, caratterizzate da 2 attributi: gid, che non è altro che un codice identificativo, e the_geom, cioè un campo contenente una stringa alfanumerica che codifica, in ambito GIS, la posizione in cui si trova quel determinato elemento. Queste 2 ultime entità vengono collegate rispettivamente a *Quadro* e *Punto Luce* tramite le associazioni *Cartografia Quadro* e *Cartografia Punto Luce*. In questo modo si ha un sistema con cui si associano ad un elemento della cartografia delle specifiche informazioni, che possono variare a seconda della natura dell'oggetto in questione, come avviene per *Posizione Quadro* e *Posizione Punto Luce*. Tale aspetto risulta evidente leggendo i requisiti, esposti in precedenza, e guardando gli attributi che si è deciso di adottare per descrivere, rispettivamente, le entità *Quadro* e *Punto Luce*.

Queste considerazioni possono essere comprese più facilmente e approfondite ulteriormente guardando il seguente schema E-R, dove sono evidenziate anche le cardinalità con cui le varie entità partecipano alle associazioni.

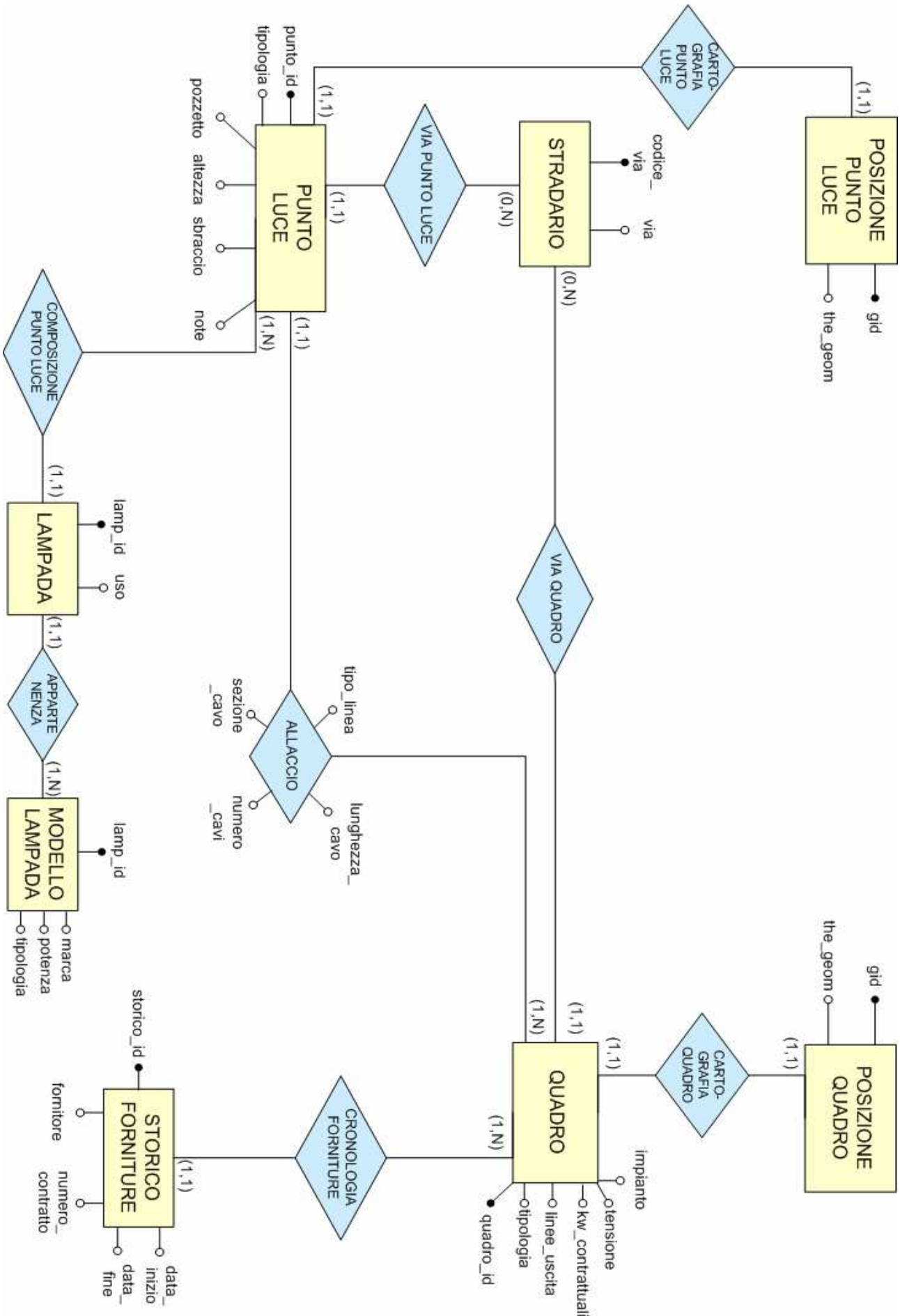


Figura 5.1 – Schema E-R proposto

Leggendo i requisiti e gli obiettivi del tirocinio, si potrebbe obiettare che non viene dato giusto risalto al concetto della linea elettrica. Infatti, ad un primo esame, sembra che gli attributi *tipo_linea*, *sezione_cavi* e *numero_cavi* di *Allaccio* potrebbero essere campi di un'entità *Linea*, che farebbe da "intermediaria", tramite la presenza di apposite associazioni, tra *Quadro* e *Punto Luce*. Ciò non è stato fatto perché sono emersi eventi della realtà da modellare che lo hanno sconsigliato: i 3 attributi menzionati non perdurano sulla medesima linea, per il semplice motivo che, per esempio, presi 2 punti luce collegati al medesimo apparato, nel primo caso la linea giunge per via aerea mentre nel secondo tramite cavidotto. Discorsi analoghi possono essere fatti per *sezione_cavi* e *numero_cavi*. Se a tutto questo aggiungiamo che fra gli scopi dell'ente non c'è quello di fornire agli uffici interessati un modo per tenere una sorta di inventario di tutte le linee, è facile comprendere il motivo della scelta presentata in Figura 5.1 .

Oltre a quanto detto, lo schema concettuale proposto ha subito delle modifiche sulla base di ulteriori aspetti evidenziati dal Ced.
In tal senso, la prima indicazione che è stata data dall'ente consiste nel togliere l'entità *Lampada*. In base a tale modifica, il numero di esemplari di un certo modello in uso presso un punto luce verrà indicato con un attributo dell'associazione *Composizione Punto Luce*. Inoltre, dato che di norma c'è un solo modello di lampada per punto luce, la molteplicità della nuova associazione è stata impostata nel seguente modo: un modello di lampada può essere montato su uno o più punti luce mentre un punto luce può montare uno o più esemplari di un solo modello di lampada.

Nel caso raro in cui si abbiano diversi modelli sullo stesso punto luce, la situazione è risolta usando l'associazione *Composizione Punto Luce* per il modello presente con il maggior numero di esemplari e avvalendosi dell'introduzione di 2 campi per mappare le "eccezioni":

- *variazioni*, dove vengono elencati i modelli di lampada presenti e non censiti tramite l'associazione e dov'è specificato se il punto luce è costituito da dei fari;

- *potenza_variazioni*, dove viene riportata la potenza totale, in Watt, delle variazioni elencate nel campo precedente.

Tale soluzione è stata scelta dall'ente perché permette un uso più agevole da parte di coloro che saranno gli utilizzatori finali del prodotto, senza perdere quelle che sono considerate informazioni fondamentali.

L'altra modifica che si è deciso di apportare alla proposta iniziale riguarda gli attributi di *Storico Forniture*, dove, dato che l'interesse è più per il semplice ordine cronologico con cui si sono susseguiti i contratti che il periodo in cui erano attivi, si è deciso di eliminare gli attributi *data_inizio* e *data_fine*. Tale ordine verrà implementato, come si vedrà meglio a livello logico, assegnando come codice identificativo un numero incrementato progressivamente. In questa logica, più alto sarà il codice identificativo assegnato e più recente sarà il contratto.

In seguito alle modifiche apportate, il nuovo schema concettuale che si è deciso di adottare viene indicato in *Figura 5.2* .

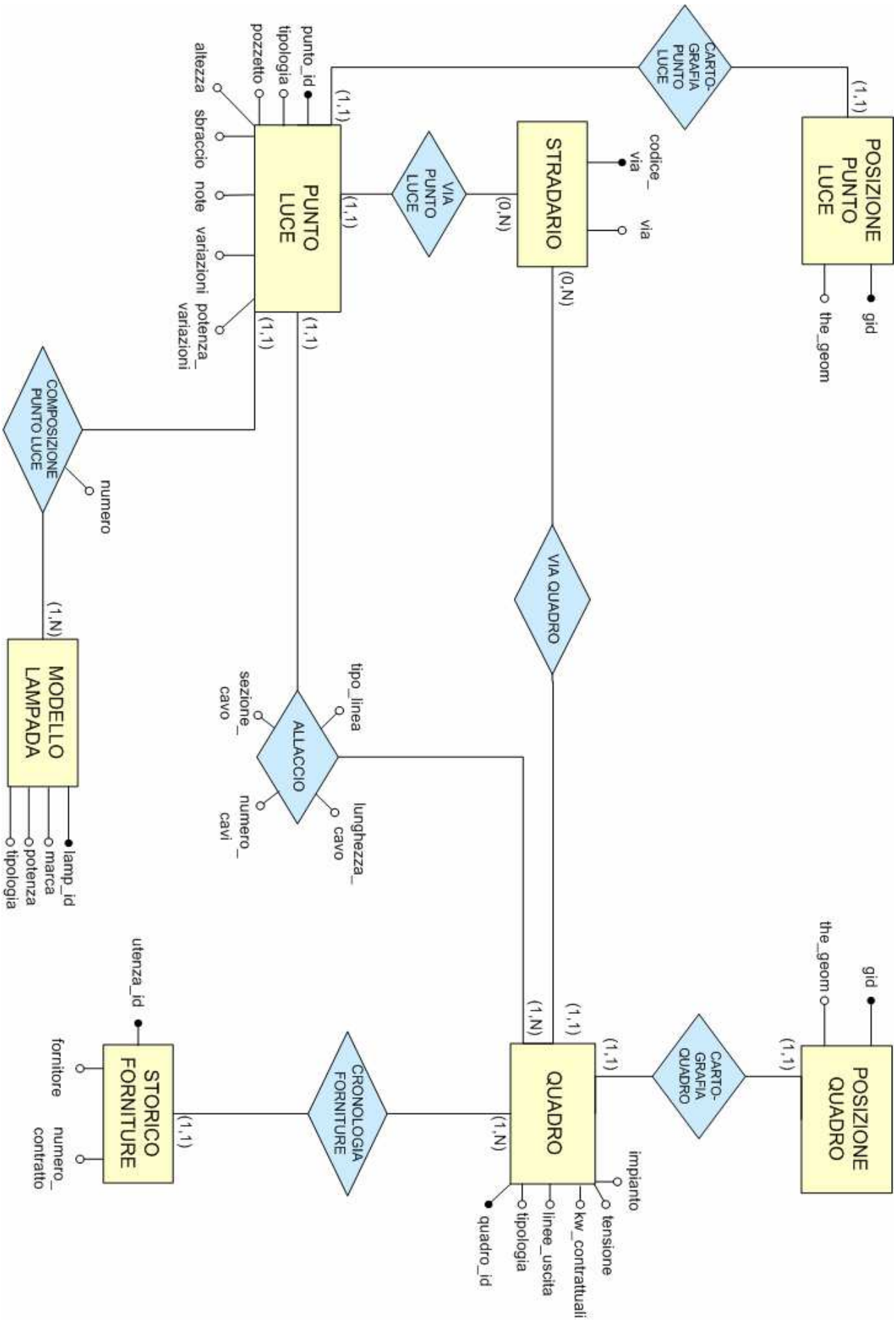


Figura 5.2 – Schema E-R adottato

5.2 SCHEMA LOGICO

Sulla base dello schema concettuale adottato (presentato in Figura 5.2) è ora possibile cominciare a progettare la struttura delle tabelle che implementeranno tale progetto.

Cominciando da *Quadro*, tale entità viene "trasformata" in una tabella con l'aggiunta di un attributo a quelli già indicati. Ciò avviene per realizzare l'associazione *Via Quadro*, che richiede la presenza di un campo *stradario_id* in cui memorizzare il codice della via in cui il quadro elettrico è situato. Inoltre, nel caso di *Cartografia Quadro*, dove la partecipazione di entrambe le entità è con molteplicità 1, si opta per l'aggiunta ai campi già riportati in *Posizione Quadro* dell'attributo *id*, che rappresenta il codice numerico del quadro a cui la specifica tupla si riferisce. Questa scelta è imposta da alcuni vincoli realizzativi, che rendono necessaria l'implementazione di *Posizione Quadro* non all'interno dello stesso Database usato per le altre tabelle. Infatti, questa verrà posizionata in quella parte dell'ambiente di *PostgreSQL/PostGIS* dedicata alla gestione della parte cartografica, il cui meccanismo, dopo essere già stato descritto nel *capitolo 4*, verrà meglio approfondito in seguito.

Proseguendo con *Punto Luce*, va detto che qui l'incremento del numero di attributi risulta abbastanza significativo. Ciò è dovuto al fatto che l'entità in questione è coinvolta in diverse associazioni uno-a-molti, dove partecipa con cardinalità 1. Per questo motivo, vale la pena analizzare ciascuna delle associazioni in cui è coinvolta, vedendo le implicazioni che ne derivano:

- *Allaccio* comporta, innanzitutto, l'aggiunta del campo *quadro_id*, a cui viene assegnato il ruolo di mantenere il codice del quadro a cui il punto luce è collegato. In aggiunta a ciò, la tabella eredita anche tutti gli attributi propri dell'associazione, cioè *lunghezza_cavo*, *sezione_cavo*, *numero_cavi* e *tipo_linea*;
- *Via Punto Luce*, che coinvolge *Punto Luce* e *Stradario*, porta all'introduzione del campo *stradario_id*, con cui si va a mappare, memorizzando l'identificativo opportuno, la via dove è situato il punto luce;
- *Composizione Punto Luce* porta con se la definizione di 2 campi: *lamp_id* e *numero_lampade*. Il primo ha il significato di indicare il modello di lampada montato, mentre il secondo quello di specificare il numero di esemplari di tale modello che sono presenti sul punto luce.

Per quanto riguarda *Cartografia Punto Luce* vale un discorso del tutto analogo a ciò che è già stato detto per *Cartografia Quadro*, traducendo il tutto con l'aggiunta di un campo *id* a *Posizione Punto Luce*.
Rimane da trattare l'associazione *Cronologia Forniture*, a cui partecipano *Storico Forniture* (con cardinalità 1) e *Quadro* (con molteplicità N). Seguendo la logica praticata fin qui, si procede inglobando fra gli attributi di *Storico Forniture* il campo *quadro_id*, contenente l'identificativo del quadro a cui quel contratto fa o ha fatto riferimento.
Lo schema logico, definito sulla base delle osservazioni appena esposte, è quello di Figura 5.3 .

Visionando attentamente tale schema è possibile notare come sia opportuno apportare alcune modifiche per renderlo ancora più rispondente a quelle che sono le buone norme di progettazione, come quelle dettate dalle teorie sulla normalizzazione. Una di queste afferma che è necessario adottare delle misure per ridurre le anomalie di inserimento o modifica e, allo stesso tempo, contenere al minimo lo spazio di memoria occupato.
Sulla base di tale regola, ci sono diversi attributi soggetti a probabili ridondanze, che accrescono il rischio di creare inconsistenze nei dati, tali da rendere del tutto inattendibile qualsiasi consultazione o statistica che si voglia fare su di essi. Con queste motivazioni, si è deciso di inserire una tabella per ciascuno dei seguenti campi:

- il campo *fornitore* della tabella *Storico Forniture*;
- *tipologia* di *Quadro*;
- *impianto* di *Quadro*;
- *tipologia* di *Punto Luce*;
- *pozzetto* di *Punto Luce*;
- *tipo_linea* di *Punto Luce*;
- *marca* di *Modello Lampada*;
- *tipologia* di *Modello Lampada*.

Il risultato di queste modifiche può essere visionato nella Figura 5.4 .

Questi correttivi potranno essere maggiormente apprezzati quando verranno predisposte le applicazioni per la modifica del Database, dove, per i campi

menzionati, sarà possibile inserire i valori scegliendoli da appositi menu a tendina.

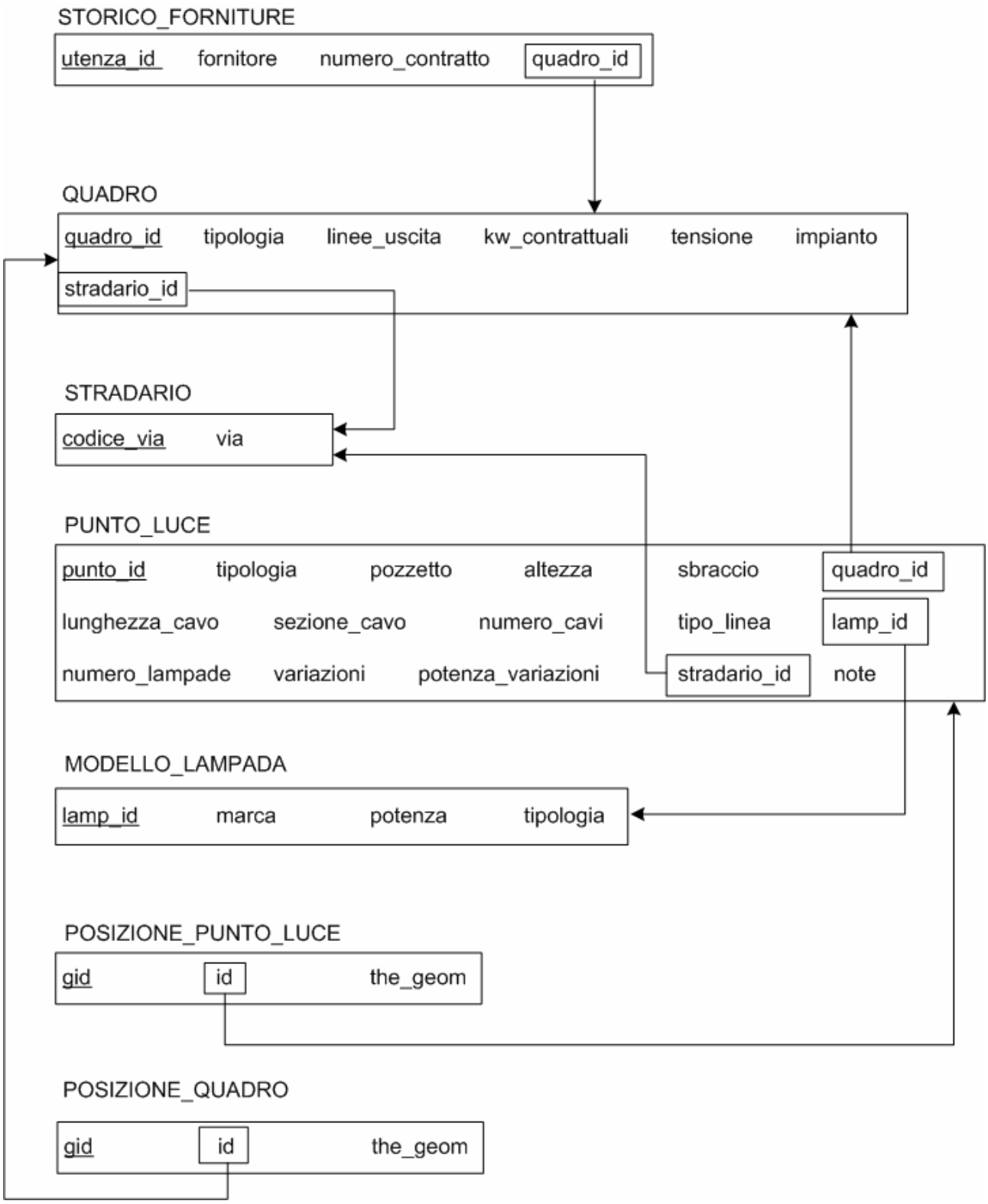


Figura 5.3 – Schema logico

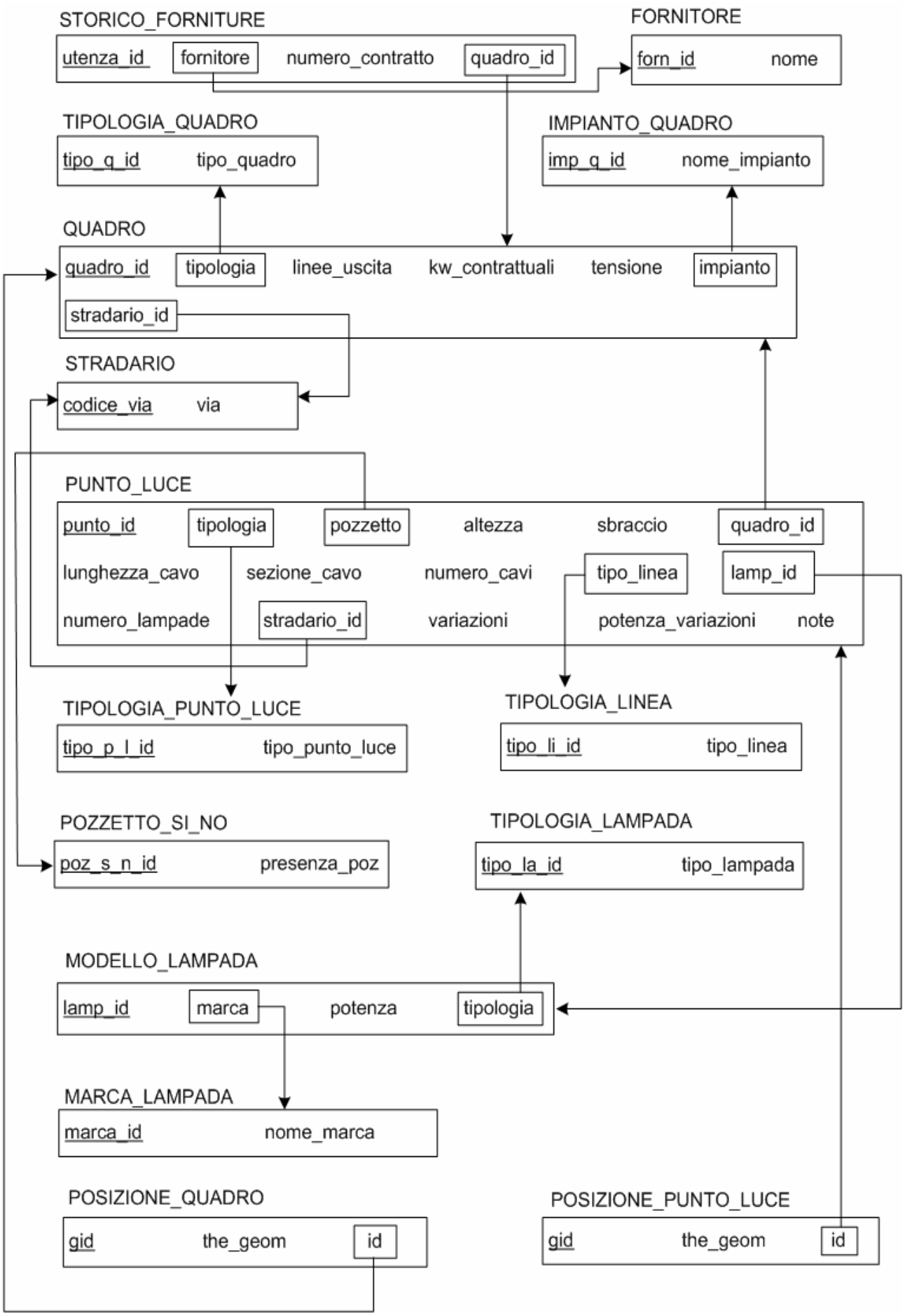


Figura 5.4 – Schema logico dopo l'introduzione dei correttivi

5.3 APPLICAZIONI PHP

Adesso, dopo aver visto le sembianze della struttura del database che verrà adottata, è venuto il momento di fare una panoramica sulle necessità che vanno soddisfatte dal punto di vista applicativo.

Quello di cui si ha bisogno è innanzitutto un sistema di autenticazione che permetta, nel presente, di gestire gli accessi alla banca dati per effettuare operazioni di modifica o consultazione e, in futuro, qualora lo si desideri, di classificare le varie utenze in appropriate classi, ciascuna rappresentante una categoria ben precisa di dipendenti.

Una volta messo appunto l'accesso al sistema, va prevista la predisposizione di quell'insieme di applicazioni incaricate di interfacciarsi direttamente con i dati.

Principalmente va definita quella parte rivolta alla gestione globale dell'inserimento/modifica dei dati. L'idea, in questo caso, consiste nel creare un qualcosa che permetta di individuare, all'interno di una singola schermata, l'elenco dei valori di un certo quadro, assieme ad una parte sottostante dove siano visionabili e modificabili le informazioni sui vari punti luce che vi sono agganciati. Nell'occasione si intende aggiungere 2 campi che, a differenza degli altri, non sono ne modificabili ne parte integrante del Database:

- *potenza totale* (nella parte contenente i dati sui *punti luce*), dove viene fatto il calcolo automatico della potenza utilizzata dal singolo punto luce;
- *potenza consumata* (nella parte riferita al *quadro elettrico*), dove si calcolano, in contrapposizione al campo *kw contrattuali*, i Watt che vengono effettivamente utilizzati dall'insieme dei punti luce che si agganciano ad un certo quadro.

Tale schermata sarà una delle 2 voci di un menu a schede, dove verrà inserita anche una parte che permette, una volta selezionato un certo quadro, l'inserimento nello storico di tutti i contratti che sono stati stipulati per quel elemento della rete. Di seguito, per agire sulle tabelle *Fornitore*, *Tipologia_Quadro*, *Impianto_Quadro*, *Stradario*, *Tipologia_Punto_Luce*, *Tipologia_Linea*, *Pozzetto_Si_No*, *Tipologia_Lampada*, *Modello_Lampada* e *Marca_Lampada* si è disposta l'implementazione di un meccanismo che permetta, a partire dalle parti già descritte e tramite le relazioni evidenziate nella *Figura 5.4*, di entrare in apposite applicazioni che rendano possibile l'inserimento di nuove voci nelle tabelle appena elencate. A questo punto della trattazione è possibile apprezzare con maggiore chiarezza il

perché del passaggio dallo schema logico di *Figura 5.3* a quello di *Figura 5.4* . Infatti, con questa struttura è possibile per l'utilizzatore selezionare il valore di certi campi da un menu a tendina e, se il valore che si vuole inserire non è già presente, entrare in un'apposita applicazione e inserirlo in pianta stabile, permettendone l'utilizzo anche in fase di successive immissioni. Tale procedura, inoltre, ha il pregio di gestire, in modo del tutto trasparente all'utilizzatore, l'assegnazione dell'appropriato valore numerico a ciascun campo: per esempio, supposto che la tabella *Fornitore* contenga una tupla per Enel (con identificativo 1) e una per Eni (2), nella schermata dei dati sullo storico delle forniture l'utilizzatore potrà selezionare per il campo *fornitore* una delle 2 voci dal menu a tendina e l'applicazione si occuperà, in modo completamente automatico, di inserire il codice identificativo corrispondente.

Infine, alla parte rivolta alla manipolazione dei dati si è deciso di aggiungere anche una trattazione squisitamente consultiva. In tal senso, si è pianificata la creazione di applicazioni per soddisfare 3 scopi:

- visualizzare i vari punti luce, raggruppati per quadro elettrico;
- visionare tutti i contratti stipulati nel tempo, raggruppati per quadro elettrico;
- avere l'elenco di tutti i quadri elettrici con i vari campi, a cui se ne aggiunge uno dove si elencano tutte le vie "alimentate" da quel determinato quadro.

La divisione delle applicazioni in queste 2 categorie risponde all'esigenza, di vitale importanza, di soddisfare eventuali distinzioni degli utenti in varie categorie, operabile in fase di autenticazione.

6. SVILUPPO

6.1 DEFINIZIONE SCHEMA LOGICO

Di seguito viene riportato il codice SQL che è stato utilizzato per definire lo schema logico esposto nella Figura 5.4 .

STORICO_FORNITURE

```
CREATE TABLE storico_forniture (
      utenza_id integer PRIMARY KEY,
      fornitore integer,
      numero_contratto character varying(20),
      quadro_id integer
);

CREATE SEQUENCE storico_forniture_utenza_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

FORNITORE

```
CREATE TABLE fornitore (
      forn_id integer PRIMARY KEY,
      nome character varying(50)
);

CREATE SEQUENCE fornitore_forn_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

TIPOLOGIA_QUADRO

```
CREATE TABLE tipologia_quadro (
      tipo_q_id integer PRIMARY KEY,
      tipo_quadro character varying(50)
);

CREATE SEQUENCE tipologia_quadro_tipo_q_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

IMPIANTO_QUADRO

```
CREATE TABLE impianto_quadro (
      imp_q_id integer PRIMARY KEY,
      nome_impianto character(50)
);

CREATE SEQUENCE impianto_quadro_imp_q_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

QUADRO

```
CREATE TABLE quadro (
      quadro_id integer PRIMARY KEY,
      tipologia integer,
      linee_uscita integer,
      kw_contrattuali integer,
      tensione integer,
      impianto integer,
      stradario_id integer
);

CREATE SEQUENCE quadro_quadro_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

PUNTO_LUCE

```
CREATE TABLE punto_luce (
      punto_id integer PRIMARY KEY,
      tipologia integer,
      pozzetto integer,
      altezza integer,
      sbraccio integer,
      quadro_id integer,
      lunghezza_cavo integer,
      sezione_cavo integer,
      numero_cavi integer,
      tipo_linea integer,
      lamp_id integer,
      numero_lampade integer
      stradario_id integer,
      variazioni character varying(200),
      potenza_variazioni integer,
      note character varying(500),
);
CREATE SEQUENCE punto_luce_punto_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

TIPOLOGIA_PUNTO_LUCE

```
CREATE TABLE tipologia_punto_luce (
      tipo_p_l_id integer PRIMARY KEY,
      tipo_punto_luce character varying(30)
);

CREATE SEQUENCE tipologia_punto_luce_tipo_p_l_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

TIPOLOGIA_LINEA

```
CREATE TABLE tipologia_linea (
      tipo_li_id integer PRIMARY KEY,
      tipo_linea character varying(30)
);

CREATE SEQUENCE tipologia_linea_tipo_li_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

POZZETTO_SI_NO

```
CREATE TABLE pozzetto_si_no (
      poz_s_n_id integer PRIMARY KEY,
      presenza_poz character(2)
);

CREATE SEQUENCE pozzetto_si_no_poz_s_n_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

TIPOLOGIA_LAMPADA

```
CREATE TABLE tipologia_lampada (
      tipo_la_id integer PRIMARY KEY,
      tipo_lampada character varying(20)
);

CREATE SEQUENCE tipologia_lampada_tipo_la_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

MODELLO_LAMPADA

```
CREATE TABLE modello_lampada (
      lamp_id integer PRIMARY KEY,
      marca integer,
      potenza integer,
      tipologia integer
);

CREATE SEQUENCE modello_lampada_lamp_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

MARCA_LAMPADA

```
CREATE TABLE marca_lampada (
      marca_id integer NOT NULL,
      nome_marca character varying(30)
);

CREATE SEQUENCE marca_lampada_marca_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

CHIAVI_ESTERNE

```
ALTER TABLE ONLY storico_forniture
ADD CONSTRAINT storico_forniture_fornitore_fkey FOREIGN KEY
(fornitore) REFERENCES fornitore(forn_id) ON UPDATE CASCADE ON
DELETE RESTRICT;

ALTER TABLE ONLY storico_forniture
ADD CONSTRAINT storico_forniture_quadro_id_fkey FOREIGN KEY
(quadro_id) REFERENCES quadro(quadro_id) ON UPDATE RESTRICT ON
DELETE RESTRICT;

ALTER TABLE ONLY quadro
ADD CONSTRAINT quadro_tipologia_fkey FOREIGN KEY (tipologia)
REFERENCES tipologia_quadro(tipo_q_id) ON UPDATE CASCADE ON DELETE
RESTRICT;

ALTER TABLE ONLY quadro
ADD CONSTRAINT quadro_impianto_fkey FOREIGN KEY (impianto)
REFERENCES impianto_quadro(imp_q_id) ON UPDATE CASCADE ON DELETE
RESTRICT;

ALTER TABLE ONLY punto_luce
ADD CONSTRAINT punto_luce_lamp_id_fkey FOREIGN KEY (lamp_id)
REFERENCES modello_lampada(lamp_id) ON UPDATE CASCADE ON DELETE
RESTRICT;
```

```
ALTER TABLE ONLY punto_luce
ADD CONSTRAINT punto_luce_pozzetto_fkey FOREIGN KEY (pozzetto)
REFERENCES pozzetto_si_no(poz_s_n_id) ON UPDATE CASCADE ON DELETE
RESTRICT;

ALTER TABLE ONLY punto_luce
ADD CONSTRAINT punto_luce_quadro_id_fkey FOREIGN KEY (quadro_id)
REFERENCES quadro(quadro_id) ON UPDATE CASCADE ON DELETE
RESTRICT;

ALTER TABLE ONLY punto_luce
ADD CONSTRAINT punto_luce_tipo_linea_fkey FOREIGN KEY
(tipo_linea) REFERENCES tipologia_linea(tipo_li_id) ON UPDATE
CASCADE ON DELETE RESTRICT;

ALTER TABLE ONLY punto_luce
ADD CONSTRAINT punto_luce_tipologia_fkey FOREIGN KEY (tipologia)
REFERENCES tipologia_punto_luce(tipo_p_l_id) ON UPDATE CASCADE ON
DELETE RESTRICT;

ALTER TABLE ONLY modello_lampada
ADD CONSTRAINT modello_lampada_marca_fkey FOREIGN KEY (marca)
REFERENCES marca_lampada(marca_id) ON UPDATE CASCADE ON DELETE
RESTRICT;

ALTER TABLE ONLY modello_lampada
ADD CONSTRAINT modello_lampada_tipologia_fkey FOREIGN KEY
(tipologia) REFERENCES tipologia_lampada(tipo_la_id) ON UPDATE
CASCADE ON DELETE RESTRICT;
```

Per ciò che riguarda *Stradario* si deve specificare che una tabella contenente tali informazioni esisteva già, visto che l'esigenza di gestire questo tipo di dati era già sorta nella realizzazione di altri database. Per questo motivo, utilizzando le funzionalità offerte da *dblink*, si è deciso di definire nel DB sui punti luce la seguente vista, da cui le varie tabelle possono soddisfare i propri bisogni informativi:

STRADARIO

```
CREATE VIEW stradario AS

SELECT stradario."CodiceVia", stradario."Via",
stradario."unica/sinistra min", stradario."sinistra max",
stradario."destra min", stradario."destra max", stradario."CAP",
stradario."FRAZIONE"
FROM dblink('dbname=dominio_comunale host=xx.xx.xx.xx
user=postgres password=pippo'::text,
'SELECT "CodiceVia", "Via", "unica/sinistra min", "sinistra max",
"destra min", "destra max", "CAP", "FRAZIONE" FROM
stradario'::text)
stradario("CodiceVia" numeric, "Via" character varying,
"unica/sinistra min" character varying, "sinistra max" character
varying, "destra min" character varying, "destra max" character
varying, "CAP" numeric, "FRAZIONE" character varying);
```

In quest'ultimo caso, si deve specificare come la scelta differisca, per i campi, da ciò che è stato indicato nello schema logico. Infatti, mentre qui si è pensato di riportare in una vista l'intera tabella presente in un altro database, nel modello logico sono stati riportati solamente quelli che sono gli attributi d'interesse per il caso in esame.

Oltre a tutto ciò, si è deciso di prevedere la presenza di alcune viste che creino una sorta di "collegamento" fra il database in questione e la parte cartografica, che verrà trattata di seguito. In particolare, tale decisione è stata presa per raggruppare, a fini consultivi, certe informazioni sui quadri elettrici e sui punti luce che dovranno essere visualizzabili direttamente sulle mappe. Di seguito viene riportato il codice delle suddette viste:

CONTRATTI_IN_ESSERE & QUADRI_MAPPA

```
CREATE VIEW contratti_in_essere AS
SELECT storico_forniture.quadro_id, fornitore.nome AS fornitore,
storico_forniture.numero_contratto
FROM storico_forniture, fornitore
WHERE ((storico_forniture.fornitore = fornitore.forn_id) AND
((storico_forniture.quadro_id, storico_forniture.utenza_id) IN
(SELECT storico_forniture.quadro_id,
max(storico_forniture.utenza_id) AS max
FROM storico_utenza
GROUP BY storico_forniture.quadro_id)));

CREATE VIEW quadri_mappa AS
SELECT quadro.quadro_id, stradario.via,
tipologia_quadro.tipo_quadro, quadro.linee_uscita,
quadro.kw_contrattuali, quadro.tensione,
impianto_quadro.nome_impianto,  contratti_in_essere.fornitore,
contratti_in_essere.numero_contratto, count(punto_luce.punto_id)
AS totale_punti_luce, sum(((modello_lampada.potenza *
punto_luce.numero_lampade) + punto_luce.potenza_variazioni)) AS
potenza_utilizzata
FROM quadro, tipologia_quadro, impianto_quadro, stradario,
punto_luce, modello_lampada, contratti_in_essere
WHERE ((((((tipologia_quadro.tipo_q_id = quadro.tipologia) AND
(impianto_quadro.imp_q_id = quadro.impianto)) AND
((quadro.stradario_id)::numeric = stradario.codicevia)) AND
(quadro.quadro_id = punto_luce.quadro_id)) AND (punto_luce.lamp_id
= modello_lampada.lamp_id)) AND (quadro.quadro_id =
contratti_in_essere.quadro_id))
GROUP BY quadro.quadro_id, stradario.via,
tipologia_quadro.tipo_quadro, quadro.linee_uscita,
quadro.kw_contrattuali, quadro.tensione,
impianto_quadro.nome_impianto,
contratti_in_essere.fornitore,
contratti_in_essere.numero_contratto
ORDER BY quadro.quadro_id;
```

PUNTI_LUCE_MAPPA

```
CREATE VIEW punti_luce_mappa AS
SELECT punto_luce.punto_id, stradario.via,
tipologia_punto_luce.tipo_punto_luce,
pozzetto_si_no.presenza_poz, punto_luce.altezza,
punto_luce.sbraccio, punto_luce.lunghezza_cavo,
punto_luce.sezione_cavo, punto_luce.numero_cavi,
tipologia_linea.tipo_linea, quadri_mappa.via AS via_quadro,
punto_luce.note, marca_lampada.nome_marca AS marca_lampada,
lampada.potenza AS potenza_lampada, punto_luce.numero_lampade,
punto_luce.variazioni, punto_luce.potenza_variazioni,
lampada.potenza * punto_luce.numero_lampade +
punto_luce.potenza_variazioni AS potenza_totale
  FROM punto_luce, tipologia_punto_luce, pozzetto_si_no,
tipologia_linea, quadri_mappa, lampada, tipologia_lampada,
marca_lampada, stradario
  WHERE punto_luce.tipologia = tipologia_punto_luce.tipo_p_l_id
AND punto_luce.pozzetto = pozzetto_si_no.poz_s_n_id AND
punto_luce.tipo_linea = tipologia_linea.tipo_li_id AND
punto_luce.quadro_id = quadri_mappa.quadro_id AND
punto_luce.lamp_id = lampada.lamp_id AND
lampada.tipologia_lampada = tipologia_lampada.tipo_la_id AND
lampada.marca = marca_lampada.marca_id AND
punto_luce.stradario_id::numeric = stradario.codicevia
  ORDER BY punto_luce.punto_id;
```

Infine, per quanto riguarda le altre 2 tabelle indicate ma di cui non è stato ancora riportato il codice, verranno viste quando si analizzeranno le operazioni necessarie per mettere in piedi tutta la parte cartografica, visto che queste parti risultano esclusivamente funzionali a tale scopo.

6.2 DEFINIZIONE APPLICAZIONI

Ora è arrivato il momento di vedere le varie schermate, già descritte nella parte sulla progettazione, che sono state create per gestire i dati contenuti nel database.
Cominciamo con la schermata principale, contenente i valori dei campi di un singolo quadro elettrico, accoppiati con l’elenco dei punti luce che vi si agganciano:

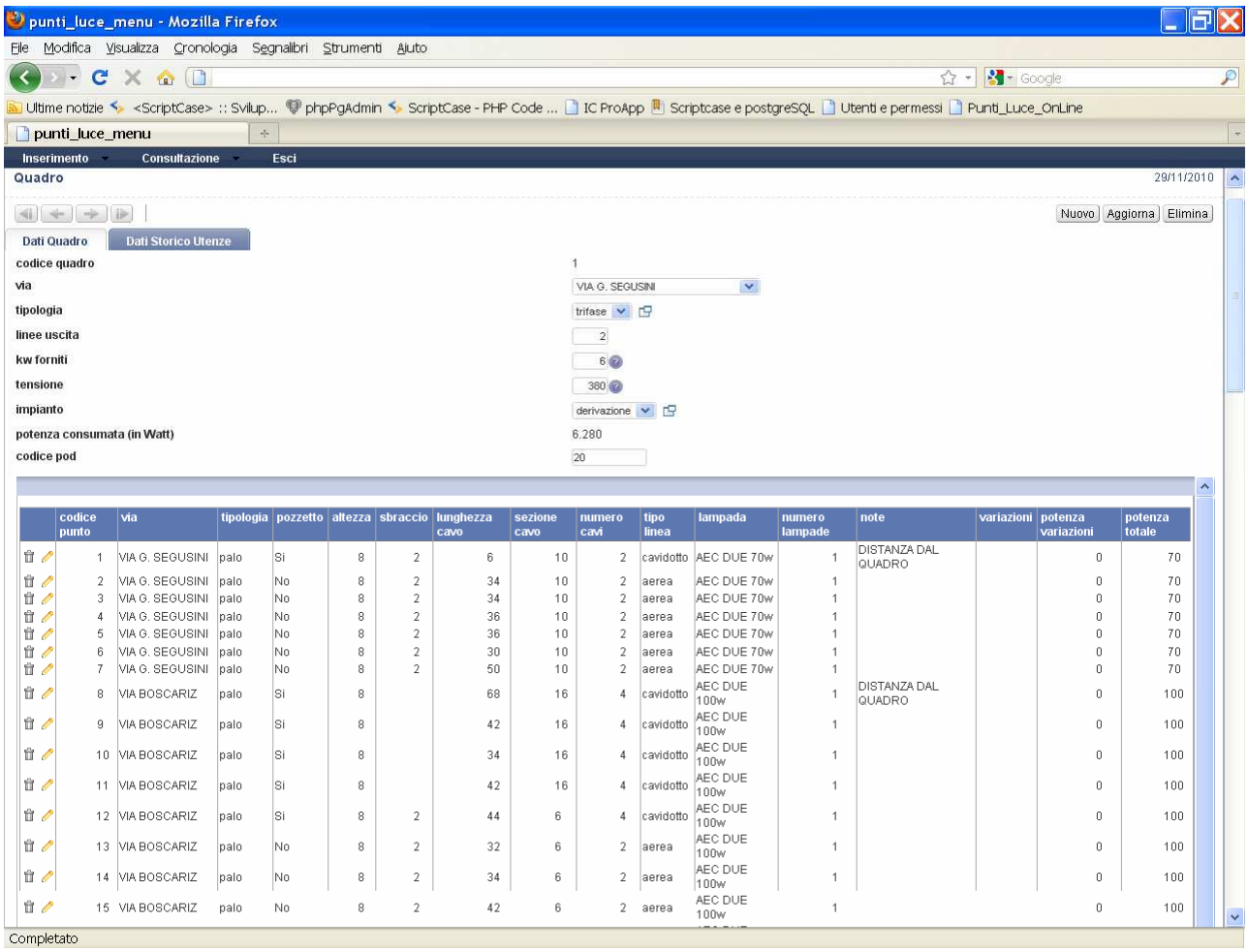


Figura 6.1 – Form Quadro – Punti Luce

Com'è possibile vedere dalla *Figura 6.1*, l'applicazione in questione è inserita in un menu a schede, da cui può essere richiamata la lista dello storico di tutti i contratti che sono stati stipulati per fornire l'energia elettrica ad uno specifico quadro:

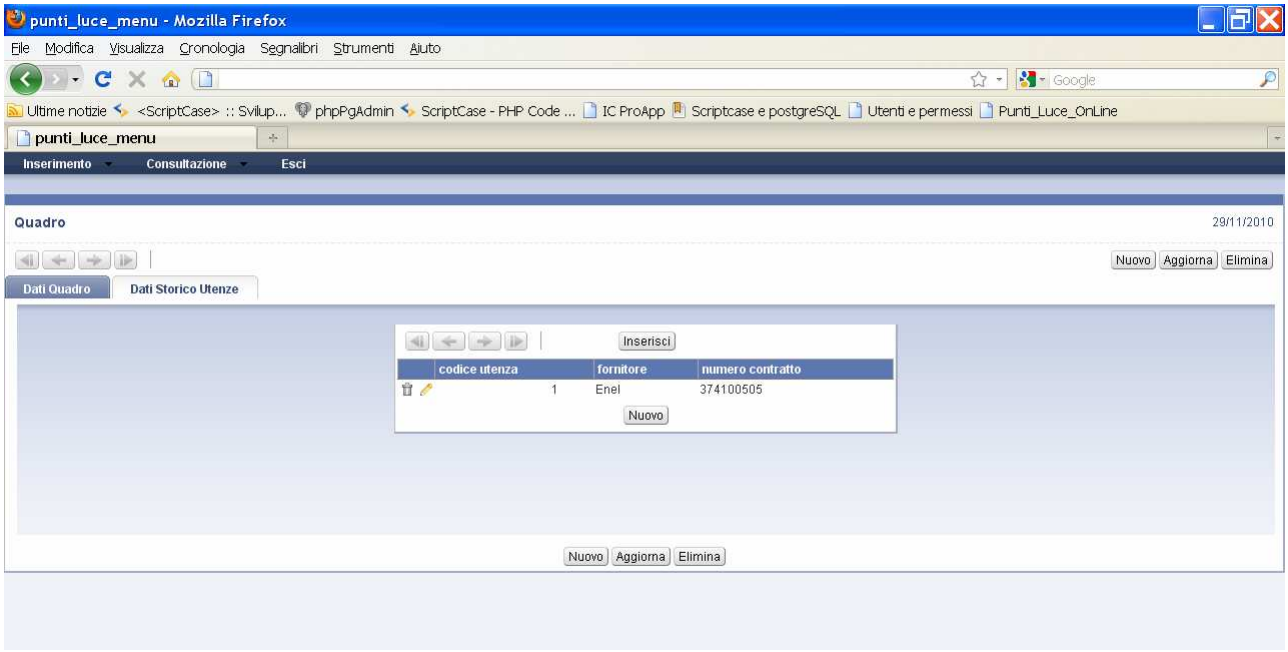


Figura 6.2 – Form per gestire i dati sullo storico forniture di ciascun quadro elettrico

È giusto dire che, per poter aprire l'applicazione che interessa in un certo momento, come quella appena mostrata, sono stati predisposti degli appositi menu a tendina. Per esempio, per accedere alle schermate appena descritte è sufficiente posizionarsi sulla voce "Inserimento" presente nella barra blu posta in alto (vedi *Figure 6.1 e 6.2*), e da qui selezionare la voce "Quadro".

Sotto la medesima voce si trova "Lampada", che permette di aprire una videata in cui è possibile consultare ed inserire i vari modelli di lampade in uso nella rete. La *Figura 6.3*, oltre a illustrare la form che è stata definita per gestire il tipo d'informazione menzionata, evidenzia anche quel meccanismo con cui vengono inseriti i dati in certi attributi.

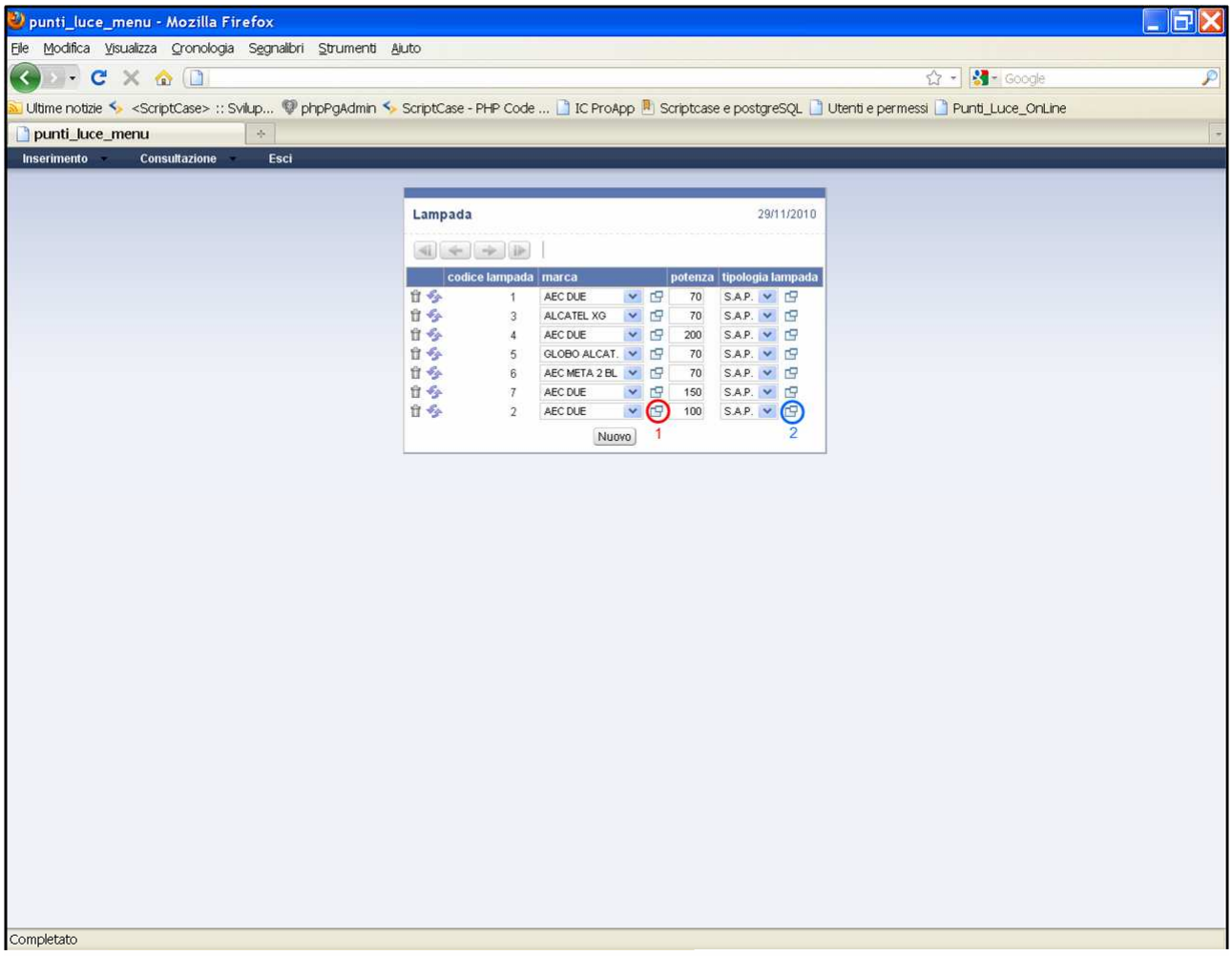


Figura 6.3 – Form per l'inserimento dei modelli di lampade

Infatti nei campi "marca" e "tipologia" i valori vengono inseriti o scegliendoli da un menu a tendina o, nel caso in cui si voglia aggiungere una scelta non già presente

nella lista, cliccando sulle icone evidenziate in *Figura 6.3* . In tal caso si aprirà un'apposita finestra con cui sarà possibile immettere una nuova scelta. Essendo che dietro ad ogni elenco esiste una tabella (vedi *Marca_Lampada* e *Tipologia_Lampada* nella *Figura 5.4*), con questa impostazione è possibile per l'utente riempire i campi in questione senza dover conoscere gli identificativi delle varie voci presenti nelle tabelle in questione. Infatti, va ricordato che ciò che viene memorizzato nel campo "marca" non è direttamente il valore selezionato dall'elenco ma un numero intero, che è il codice identificativo che viene associato a tale dato nella tabella *Marca_Lampada*.

In riferimento alle applicazioni rivolte alla semplice e pura visione delle informazioni, esse possono essere mandate in esecuzione selezionandole dal menu "Consultazione". Viste le necessità, si è pensato di prevedere 3 casi:

- *Quadro-Punti_Luce*, orientato a soddisfare l'esigenza di avere la lista di tutti i punti luce, raggruppati per quadro elettrico (*Figura 6.4*);

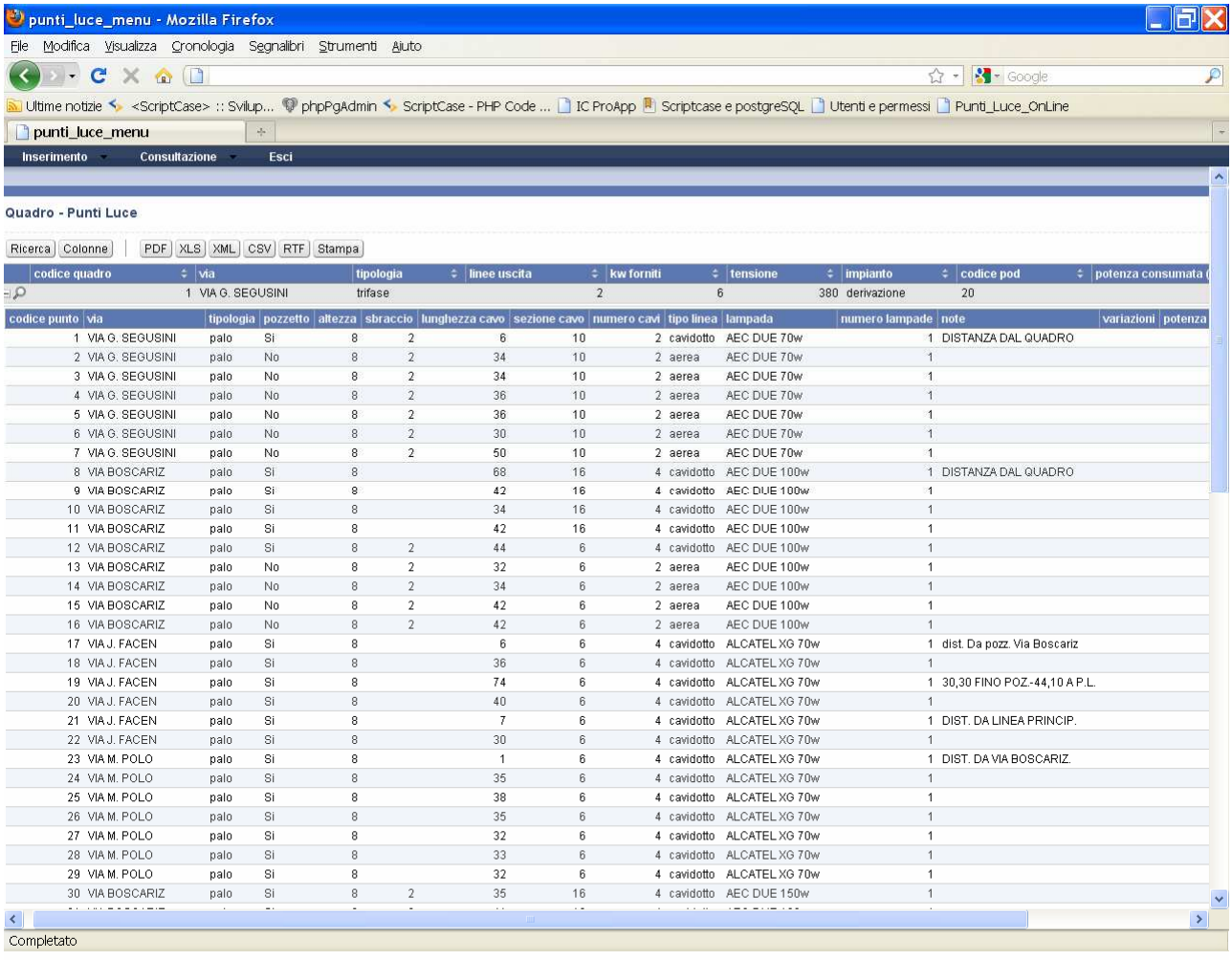


Figura 6.4 – Grid contenente i punti luce raggruppati per quadro elettrico

- *Quadro-Utenze*, dove si elencano tutti i quadri elettrici, a cui si associano i vari contratti per la fornitura di energia che sono stati stipulati negli anni (*Figura 6.5*);
- *Quadro-Vie_Alimentate*, definito con lo scopo di elencare gli attributi dei quadri elettrici, con l'aggiunta di un campo contenente la lista delle vie in cui sono presenti punti luce collegati a quel quadro (*Figura 6.6*).

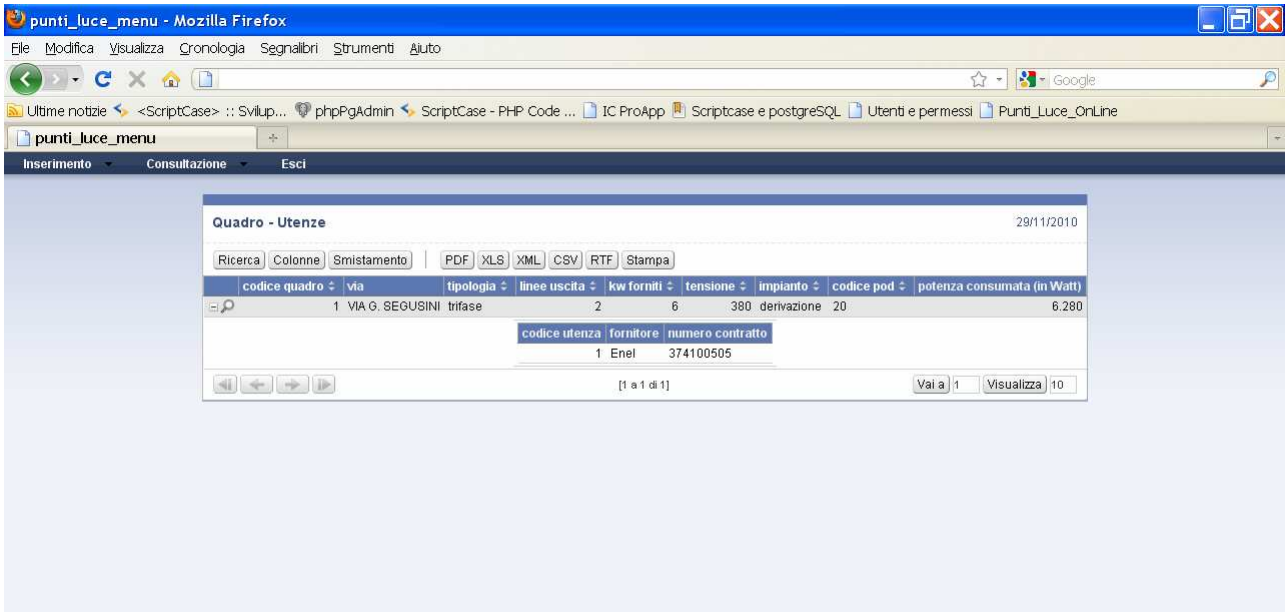


Figura 6.5 – Grid dove lo storico delle forniture viene raggruppato per quadro elettrico

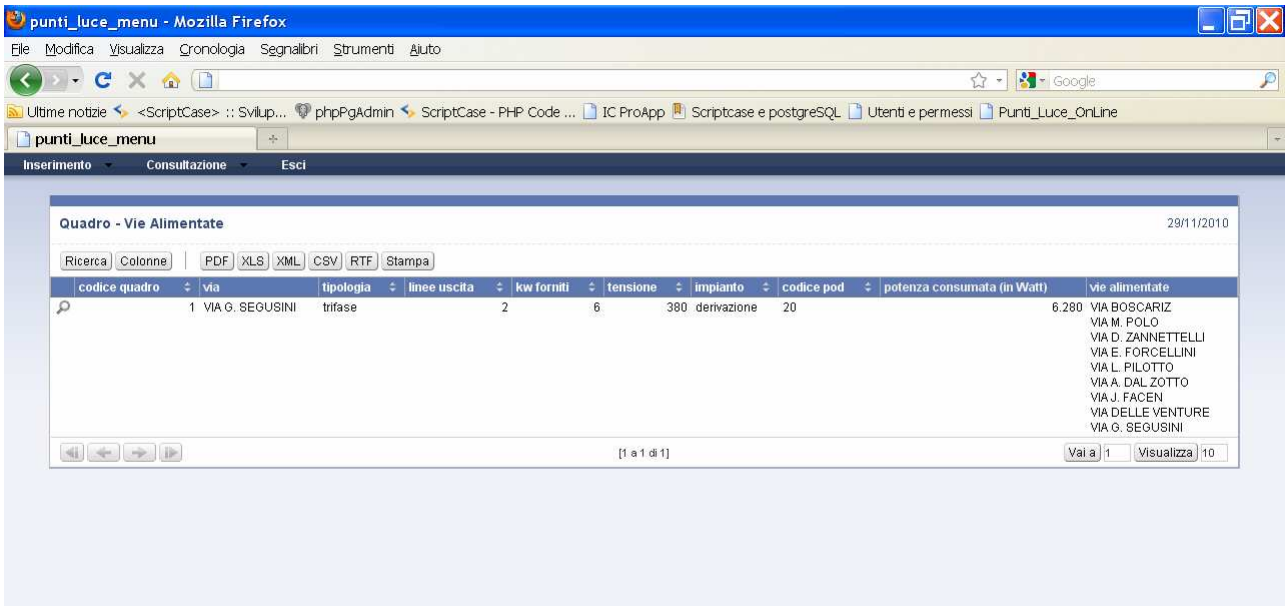


Figura 6.6 – Grid in cui si elencano le varie vie che ogni quadro alimenta

Infine, è il caso di accennare a come viene gestita tutta la parte riguardante l'autenticazione. Prima di partire con la trattazione si deve specificare che da tempo

il Comune di Feltre si è dotato di un apposito database rivolto a gestire le credenziali d'accesso alle varie banche dati. Fissato ciò, si procede con la creazione di un'apposita applicazione, contente una form dove l'utente deve inserire l'username e la password con cui può essere identificato dal sistema. Arrivati a questo punto, si predispone un codice che fissa i passaggi che il programma deve effettuare in successione:

```
sc_lookup(dataset, "SELECT utente_id, nomeutente, utente_genere
    FROM public.utente
    WHERE nomeutente = '{user}' AND pwd = '{password}'");

if ({dataset[0][0]} <= 0) {
    sc_error_message('utente o password non valido');
}
{utente} = {dataset[0][0]};
{user_sigla} = {dataset[0][1]};
{user_genere} = {dataset[0][2]};
sc_set_global({utente});
sc_set_global({user_sigla});
sc_set_global({user_genere});
```

```
sc_lookup(dataset2, "
SELECT applicazione.nome_a
FROM public.applicazione JOIN public.credenziale AS b ON
b.applicazione_id = applicazione.applicazione_id
JOIN public.ruolo AS c ON c.ruolo_id = b.ruolo_id JOIN utenteruolo
AS d ON d.ruolo_id = c.ruolo_id
JOIN utente ON utente.utente_id = d.utente_id
WHERE utente.nomeutente = '{user_sigla}' ","permessi2");
foreach({dataset2} as $line){
        $applications = $line[0];
        sc_apl_status($applications,'on');

}
```

Il codice appena riportato è stato diviso in 2 parti non perché lo sia nella realtà, ma per il semplice fatto di distinguere le fasi di cui si occupa. Nello specifico, le operazioni che si predispongono sono le seguenti:

- nella **1^ parte** ci si occupa innanzitutto di cercare nel database delle autenticazioni, tramite un'interrogazione SQL, se esiste la tupla con i campi *nomeutente* e *pwd* uguali ai valori inseriti nella form e, nel caso in cui si trovi la corrispondenza, di inserire tale tupla nella variabile vettoriale *dataset*. Successivamente, dopo aver definito un IF che va ad appurare se esiste o meno una riga che rappresenti le credenziali inserite, si dichiarano delle variabili in cui viene scomposta l'informazione che è stata stoccata in *dataset*;
- nella **2^ parte**, invece, si va ad inserire un'interrogazione SQL che ha il compito di fornire l'elenco delle applicazioni a cui l'utente ha il diritto di

accedere. L'elenco in questione viene memorizzato in una seconda variabile *vettoriale* (chiamata *dataset2*) che, in un secondo momento, viene processata da un ciclo *foreach*. A quest'ultimo e demandata la funzione di attivare l'uso, tramite la macro *sc_apl_status*, di tutte quelle applicazioni che sono presenti nella lista memorizzata in *dataset2*.

Come è possibile immaginare, una procedura simile deve essere prevista anche in fase d'uscita. In questo caso, il codice che viene scritto è:

```
sc_lookup(dataset2, " SELECT applicazione.nome_a
FROM public.applicazione JOIN public.credenziale AS b ON
b.applicazione_id = applicazione.applicazione_id
JOIN public.ruolo AS c ON c.ruolo_id = b.ruolo_id JOIN utenteruolo
AS d ON d.ruolo_id = c.ruolo_id
JOIN utente ON utente.utente_id = d.utente_id
WHERE utente.utente_id = [utente] ");

if (FALSE === {dataset2}) {
      sc_error_message("An error occurred in access to the
database.");
}
elseif (empty({dataset2})) {
      sc_error_message("No value was returned by the bank");
}
else {
echo('OK');
}
foreach({dataset2} as $line){
        $applications = $line[0];
        sc_apl_status($applications,'off');
}

sc_redir(control);
```

Con ciò si effettuano le medesime operazioni del caso precedente, svolte però in ordine inverso. Infatti, premendo la voce "Esci" presente nella barra dei menu (vedi *Figura 6.6* o precedenti), viene eseguito questo codice che:

1. crea la lista delle applicazioni che l'utente della sessione può utilizzare;
2. svolge quella serie di controlli volti a gestire tutti quei casi particolari;
3. fa partire un ciclo che disattiva l'uso di tutti quei programmi presenti nell'elenco formato al passo 1;
4. rimanda all'applicazione *control*, che rappresenta la schermata con la form d'autenticazione e il codice presentati nella pagina precedente.

6.3 PARTE CARTOGRAFICA

Dopo aver visto tutte le misure disposte per permettere la memorizzazione e la modifica delle informazioni sulla rete di illuminazione pubblica, è arrivato il momento di capire come si procede nella rappresentazione cartografica dei vari elementi.

Per arrivare ad avere delle mappe dove è possibile visualizzare la disposizione dei quadri e dei punti luce, assieme ai dati che li caratterizzano, va eseguita la seguente procedura:

1. adoperando il pacchetto *Autocad Map*, già adottato nella soluzione precedente, vengono posizionati gli elementi sulla mappa, memorizzando, per ciascuno di essi, quello che è il loro codice identificativo (generato in automatico dal database definito all'inizio di questo capitolo);
2. i disegni fatti con *Map* vengono esportati in un file Shape, per poi essere reimportati in *PostGIS* tramite *MapStorer*. Il risultato è una tabella, come quella di *Figura 6.7*, che deve avere obbligatoriamente i campi *gid* (identificativo proprio di questa nuova tabella) e *the_geom* (coordinata geografica), più l'attributo *id* con cui viene identificato l'elemento, che è stato definito dall'operatore al passaggio 1;
3. per il meccanismo con cui funziona PostGIS, descritto nel paragrafo 4.1, il nome della tabella risultato del passaggio 2 va inserito in *geometry_columns*;
4. utilizzando le funzionalità di *MapStorer* (*Figura 6.8*), si procede alla vestizione grafica dell'informazione importata. Il risultato di questo passaggio consiste in un codice, con una sintassi ben precisa, che specifica come devono presentarsi graficamente gli elementi che sono stati introdotti nella mappa;
5. il risultato di ciò che viene fatto nel passaggio precedente viene messo in un apposito file (con estensione *.map*), che verrà posizionato all'interno della struttura di *MapServer*. Per esempio, nel caso del layer che contiene la disposizione dei punti luce, il *Mapfile* si presenta in questo modo:

```
#---------- start layer c_d530_v_punti_luce001----------

 LAYER
  NAME   "10415"
  CONNECTION    "user=postgres dbname=icpro_feltre host=localhost
password=tresql20cofl port=5432"
  CONNECTIONTYPE  postgis
  TYPE  POINT
  DATA   "the_geom from c_d530_v_punti_luce001 USING UNIQUE gid"
```

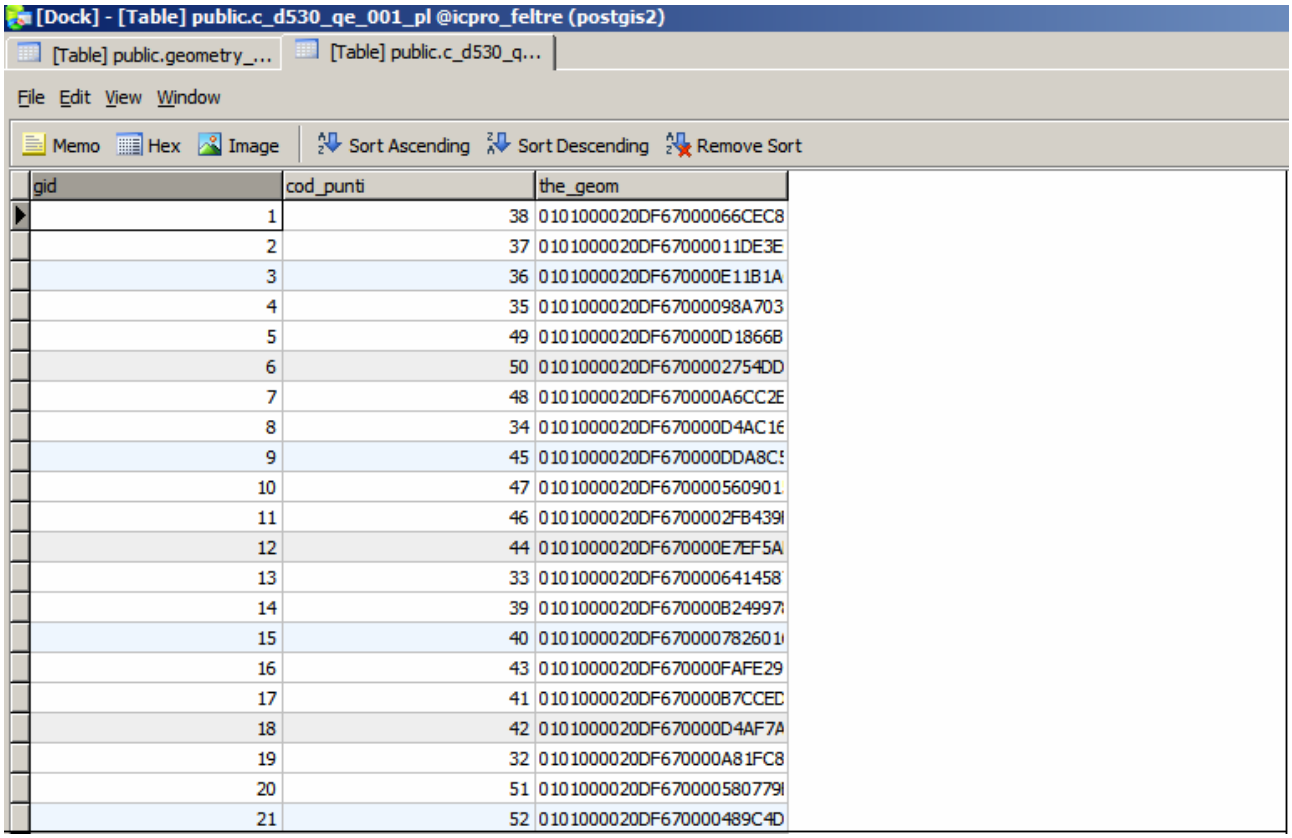
```
  STATUS  OFF

  CLASS
   NAME  "c_d530_v_punti_luce001_14781"

   STYLE
    COLOR  255 0 255
    OUTLINECOLOR  255 0 255
    SIZE  10
    SYMBOL  "circle"
   END
  END

  PROJECTION
         "init=epsg:26591"
  END

  METADATA
   ORNAME   "c_d530_v_punti_luce001"
   WMS_SRS  "epsg:26591"
   WMS_TITLE  "c_d530_v_punti_luce001"
   WMS_FEATURE_INFO_MIME_TYPE  "text/html"
  END
 END    ;
```



gid	cod_punti	the_geom
1	38	0101000020DF67000066CEC8
2	37	0101000020DF67000011DE3E
3	36	0101000020DF670000E11B1A
4	35	0101000020DF67000098A703
5	49	0101000020DF670000D1866B
6	50	0101000020DF6700002754DD
7	48	0101000020DF670000A6CC2E
8	34	0101000020DF670000D4AC16
9	45	0101000020DF670000DDA8C5
10	47	0101000020DF670000560901
11	46	0101000020DF6700002FB439I
12	44	0101000020DF670000E7EF5A
13	33	0101000020DF670000641458
14	39	0101000020DF670000B24997
15	40	0101000020DF6700007826016
16	43	0101000020DF670000FAFE29
17	41	0101000020DF670000B7CCED
18	42	0101000020DF670000D4AF7A
19	32	0101000020DF670000A81FC8
20	51	0101000020DF6700005807791
21	52	0101000020DF670000489C4D

Figura 6.7 – Tabella risultato dell'importazione di uno shape in PostGIS

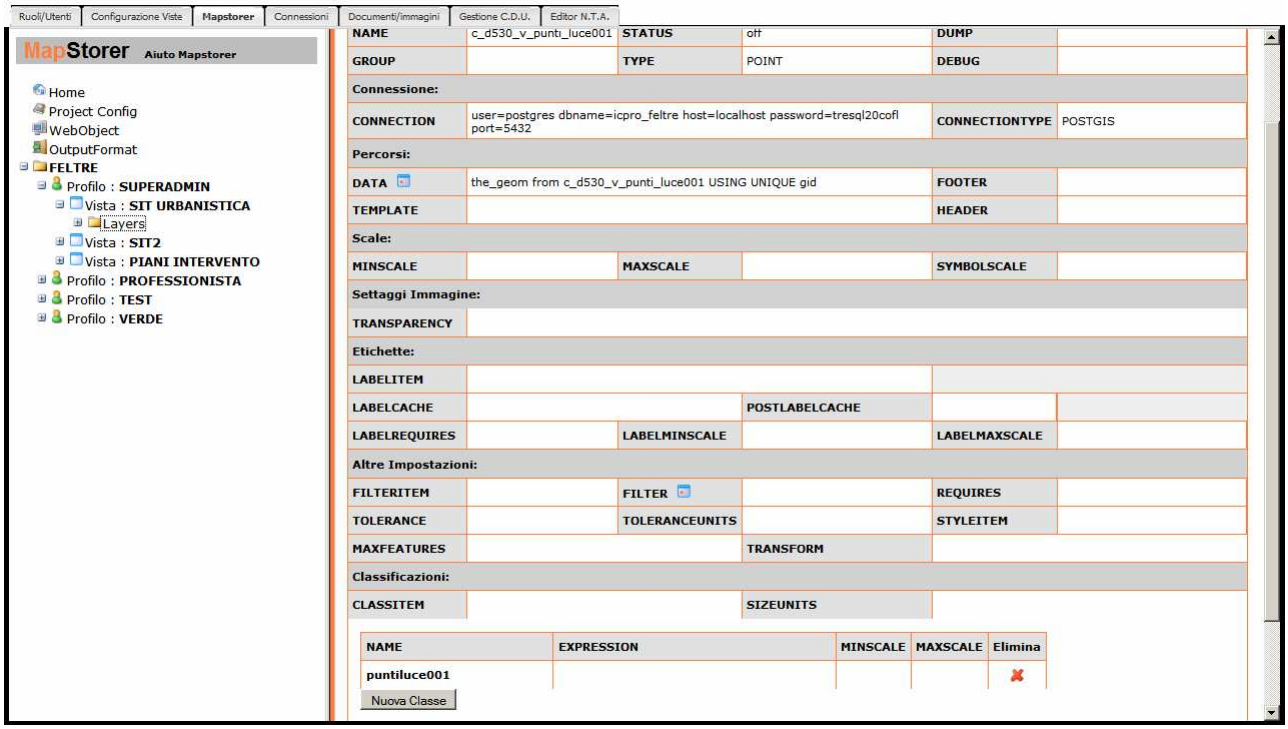


Figura 6.8 – Schermata di MapStorer con cui si fa la vestizione dell'informazione

6. sempre tramite l'utilizzo di *MapStorer*, si prosegue con la pubblicazione del layer, stabilendone quali sono le operazioni che possono essere fatte su di esso: esportazione, interrogazione, tooltip, ecc. (*Figura 6.9*);

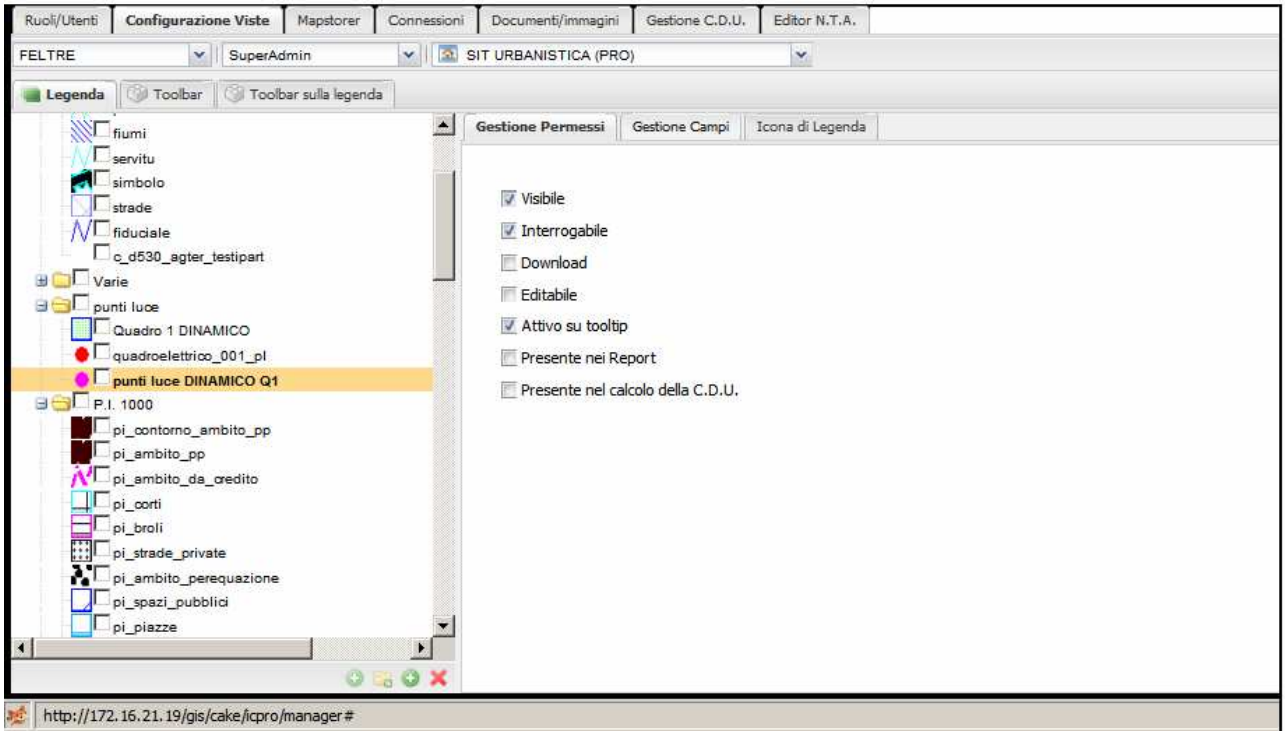


Figura 6.9 – Vista che permette di specificare le operazioni che sono abilitate su un determinato layer

7. all’interno del database cartografico vengono previste delle viste che “replicano”, tramite l’uso di dblink, *quadri_mappa* e *punti_luce_mappa*, visti nel paragrafo 6.1:

```
SELECT quadroelettrico.quadro_id, quadroelettrico.via,
quadroelettrico.tipo_quadro, quadroelettrico.linee_uscita,
quadroelettrico.kw_contrattuali, quadroelettrico.tensione,
quadroelettrico.nome_impianto, quadroelettrico.fornitore,
quadroelettrico.numero_contratto,
quadroelettrico.totale_punti_luce,
quadroelettrico.potenza_utilizzata
FROM dblink('dbname=puntiluce  host=ip.ip.ip.ip user=XXXX
password=XXX'::text, 'SELECT quadro_id,via,
tipo_quadro,linee_uscita,kw_contrattuali,tensione,nome_impiant
o, fornitore, numero_contratto,
totale_punti_luce,potenza_utilizzata from quadri_mappa'::text)
quadroelettrico(quadro_id numeric, via character varying(30),
tipo_quadro character varying(10), linee_uscita numeric,
kw_contrattuali numeric, tensione numeric, nome_impianto
character varying(30), fornitore character varying(30),
numero_contratto character varying(30), totale_punti_luce
numeric, potenza_utilizzata numeric);
```

Figura 6.10 – definizione nel db di un collegamento, via dblink, con la vista esterna contenente i dati sui quadri elettrici.

```
SELECT puntiluce.punto_id, puntiluce.via,
puntiluce.tipo_punto_luce, puntiluce.presenza_poz,
puntiluce.altezza, puntiluce.sbraccio,
puntiluce.lunghezza_cavo, puntiluce.sezione_cavo,
puntiluce.numero_cavi, puntiluce.tipo_linea,
puntiluce.via_quadro, puntiluce.note, puntiluce.marca_lampada,
puntiluce.potenza_lampada, puntiluce.numero_lampade,
puntiluce.variazioni, puntiluce.potenza_variazioni,
puntiluce.potenza_totale
FROM dblink('dbname=puntiluce  host=10.21.128.225 user=utente
password=prova'::text, 'SELECT punto_id, via, tipo_punto_luce,
presenza_poz, altezza, sbraccio, lunghezza_cavo, sezione_cavo,
numero_cavi, tipo_linea, via_quadro, note, marca_lampada,
potenza_lampada, numero_lampade, variazioni,
potenza_variazioni, potenza_totale  from
punti_luce_mappa'::text)
puntiluce(punto_id numeric, via character varying,
tipo_punto_luce character varying, presenza_poz character
varying, altezza numeric, sbraccio numeric, lunghezza_cavo
numeric, sezione_cavo numeric, numero_cavi numeric, tipo_linea
character varying, via_quadro character varying, note
character varying, marca_lampada character varying,
potenza_lampada numeric, numero_lampade numeric, variazioni
character varying, potenza_variazioni numeric, potenza_totale
numeric);
```

Figura 6.11 – definizione nel db di un collegamento, via dblink, con la vista esterna contenente i dati sui punti luce.

8. viene fatto un join fra le viste definite con il dblink e le tabelle risultato dell'importazione dello shape in PostGIS. Questo permetterà, una volta attivata la funzione *tooltip* sul layer, di poter visualizzare le informazioni di un quadro elettrico/punto luce sulla mappa, semplicemente posizionandosi con il cursore sopra la posizione in cui è collocato.

```
SELECT dblink_quadro_elettrico_punti_luce.quadro_id,
dblink_quadro_elettrico_punti_luce.via,
dblink_quadro_elettrico_punti_luce.tipo_quadro,
dblink_quadro_elettrico_punti_luce.linee_uscita,
dblink_quadro_elettrico_punti_luce.kw,
dblink_quadro_elettrico_punti_luce.tensione,
dblink_quadro_elettrico_punti_luce.nome_impianto,
dblink_quadro_elettrico_punti_luce.fornitore,
dblink_quadro_elettrico_punti_luce.numero_contratto,
dblink_quadro_elettrico_punti_luce.totale_punti_luce,
dblink_quadro_elettrico_punti_luce.potenza_utilizzata,
c_d530_qe001_qe.gid, c_d530_qe001_qe.cod_qe,
c_d530_qe001_qe.the_geom
FROM dblink_quadro_elettrico_punti_luce, c_d530_qe001_qe
WHERE ((c_d530_qe001_qe.cod_qe)::numeric =
dblink_quadro_elettrico_punti_luce.quadro_id);
```

Figura 6.12 - unisce i dati alfanumerici del quadro elettrico con la corrispettiva tabella cartografica.

```
SELECT dblink_puntiluce001.punto_id, dblink_puntiluce001.via,
dblink_puntiluce001.tipo_punto_luce,
dblink_puntiluce001.presenza_poz, dblink_puntiluce001.altezza,
dblink_puntiluce001.sbraccio,
dblink_puntiluce001.lunghezza_cavo,
dblink_puntiluce001.sezione_cavo,
dblink_puntiluce001.numero_cavi,
dblink_puntiluce001.tipo_linea, dblink_puntiluce001.via_quadro,
dblink_puntiluce001.note, dblink_puntiluce001.marca_lampada,
dblink_puntiluce001.potenza_lampada,
dblink_puntiluce001.numero_lampade,
dblink_puntiluce001.variazioni,
dblink_puntiluce001.potenza_variazioni,
dblink_puntiluce001.potenza_totale, c_d530_qe_001_pl.gid,
c_d530_qe_001_pl.cod_punti, c_d530_qe_001_pl.the_geom
FROM dblink_puntiluce001, c_d530_qe_001_pl
WHERE (dblink_puntiluce001.punto_id =
(c_d530_qe_001_pl.cod_punti)::numeric);
```

Figura 6.13 – unisce i dati alfanumerici dei punti luce con la corrispettiva tabella cartografica.

Essendo che i dati che compariranno tramite il *tooltip* sono pescati di volta in volta dal database esterno, definito all'inizio di questo capitolo, ad ogni modifica non c'è bisogno di alcun intervento sul lato cartografico. Questo è possibile perché l'informazione, tramite la tecnica appena decritta, si "propaga" automaticamente, rendendo lo shape dinamico.

7. RISULTATI

Alla fine di questo percorso è il caso di soffermarci su quelli che sono gli obiettivi che sono stati raggiunti.

Quello che si è ottenuto è un sistema che permette, a tutti coloro che ricoprono incarichi direttamente collegati a questa tematica, di avere uno strumento caricato sulla Intranet comunale, consultabile utilizzando un semplice browser web, che fornisca la posizione esatta dei vari elementi che costituiscono la rete di illuminazione pubblica (*Figura 7.1*). Inoltre, ogni qualvolta che un utente si posiziona con il cursore su uno di questi elementi, appare una piccola finestra in cui sono riportati tutti i dati che descrivono, in modo più significativo, l'oggetto (*Figure 7.2 e 7.3*).

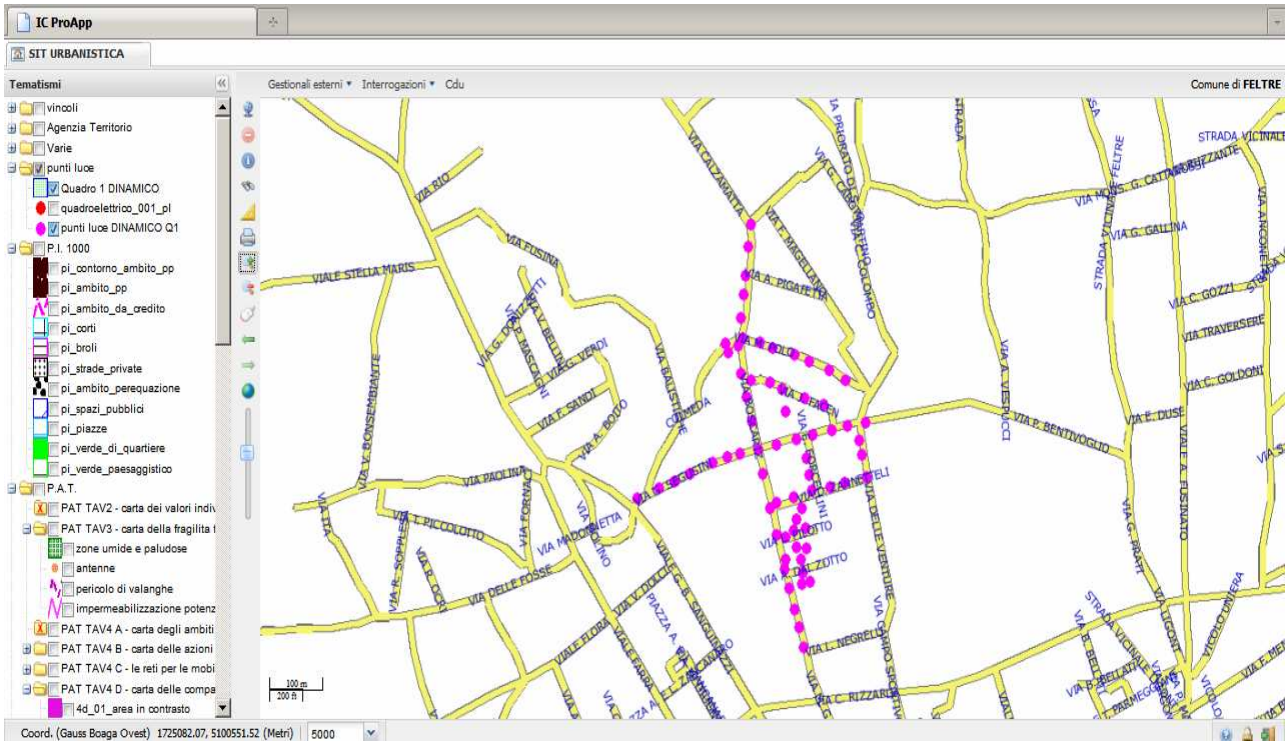


Figura 7.1 – Mappa in cui viene indicata la disposizione dei punti luce

Inoltre, una volta all'interno di questa applicazione, tramite un collegamento ad un'applicazione esterna, possono essere lanciati quei programmi che permettono, successivamente all'autenticazione, di modificare e consultare il contenuto del database.

Quest'ultimo aspetto, agganciato alla funzionalità tooltip, assume grande rilevanza all'interno del risultato finale. Infatti, una volta modificati i dati contenuti nel database sviluppato al *Paragrafo 6.1*, non è necessario apportare nessuna modifica al lato

cartografico al fine di mantenere le informazioni aggiornate.

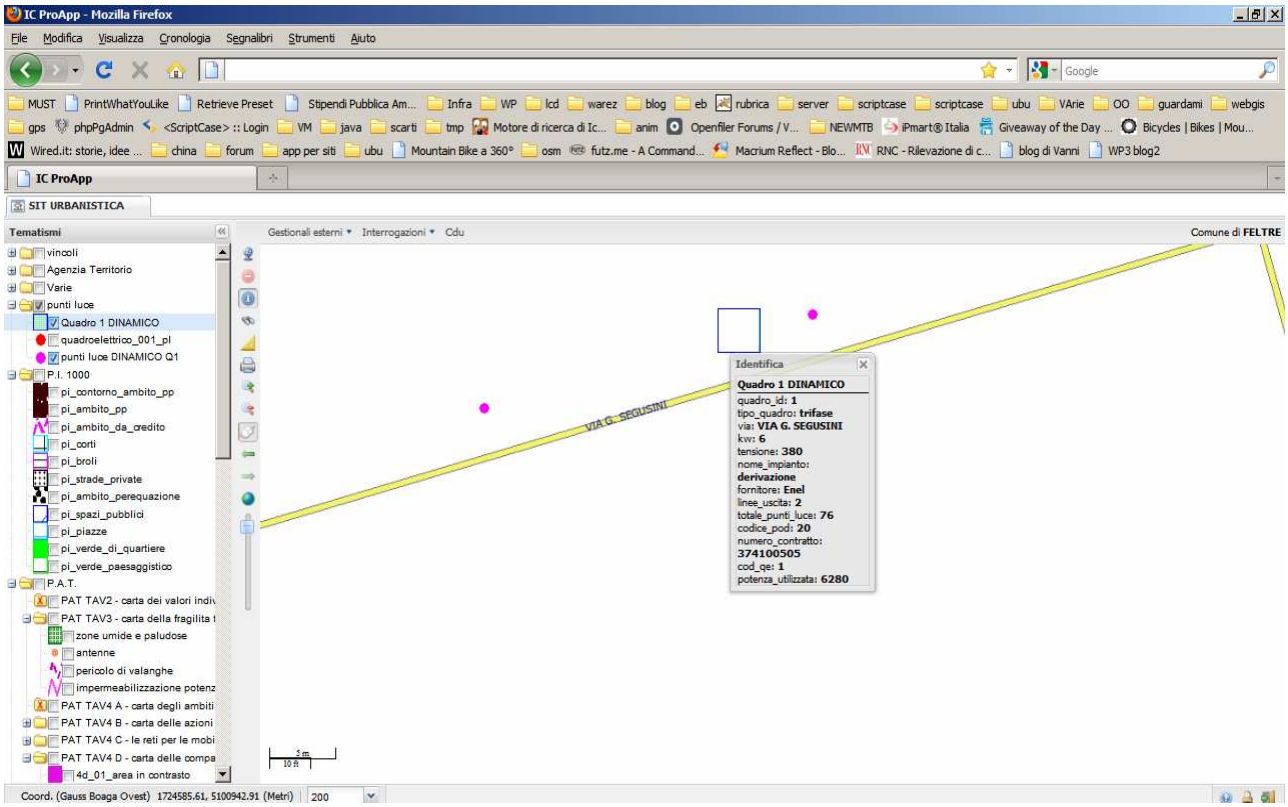


Figura 7.2 – Funzione tooltip attivata sul layer dei quadri elettrici

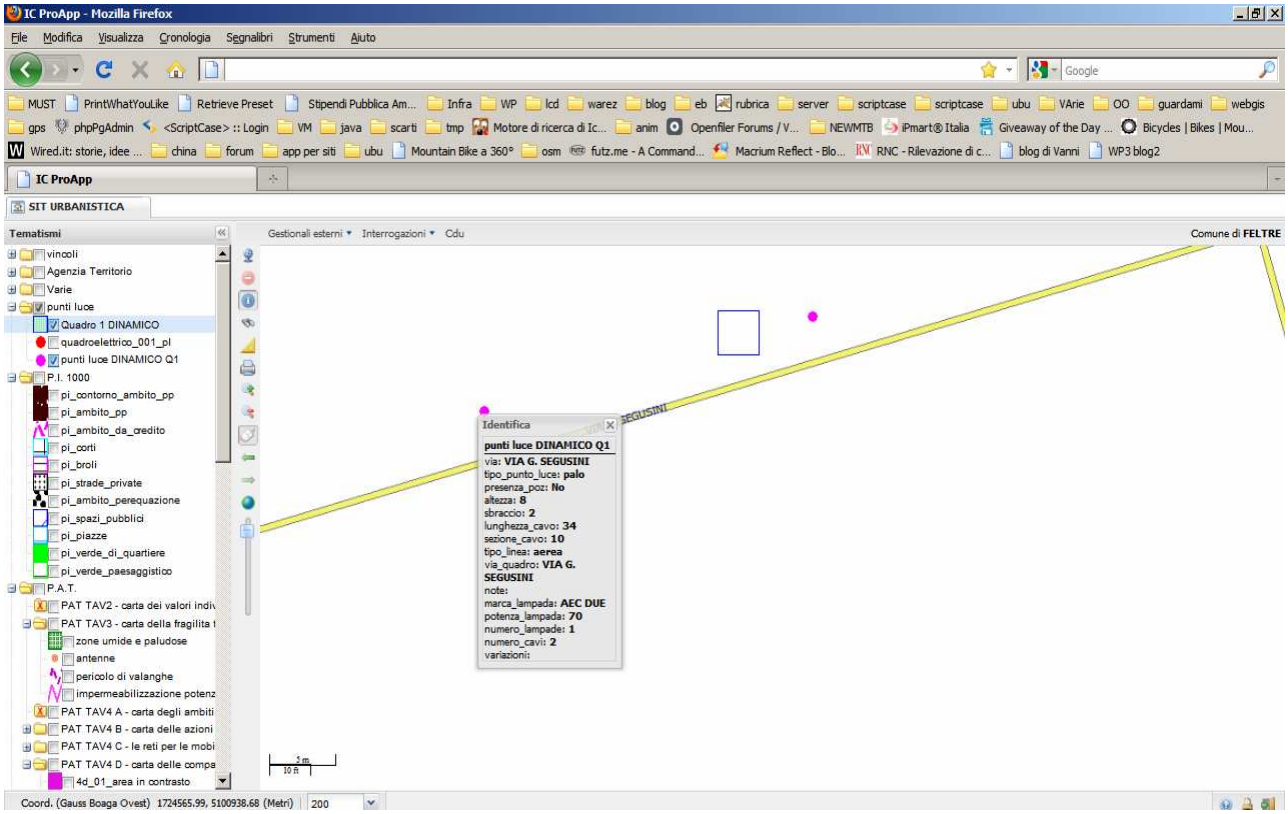


Figura 7.3 – Funzione tooltip attivata sul layer dei punti luce

Ciò risulta essere importante per 2 motivi:

- si evita che si verifichino problemi di inconsistenza dei dati;
- non è richiesto l'intervento, nell'attività di modifica dei dati dei punti luce o dei quadri elettrici già presenti sulle mappe, di un operatore specializzato che sappia come muoversi all'interno dell'ambiente cartografico.

Infine, l'ultima miglioria che è stata apportata con l'adozione di questa soluzione sta nel modo in cui vengono trattati i dati a disposizione. Questo è dovuto al fatto che, in precedenza, le informazioni venivano gestite dall'ufficio dei lavori pubblici con questi strumenti:

- *Autocad Map*, usato solamente per posizionare i vari elementi sulle mappe;
- un *foglio elettronico* per ogni quadro elettrico dove, oltre ad essere riportati i dati di tale elemento, venivano elencati tutti i valori riguardanti i punti luce che vi erano collegati.

Questo, considerando che le 2 parti erano totalmente slegate fra loro e memorizzate in locale, comportava alti rischi nel mantenimento della consistenza dei dati. In tal senso, il nuovo sistema che è stato adottato consente di eliminare questi potenziali pericoli, prevedendo:

- la gestione delle informazioni tramite una serie di tabelle opportunamente collegate;
- la messa in rete delle banche dati, tali da renderle "uniche" e consultabili da tutti quei soggetti che ne abbiano bisogno;
- uno stretto collegamento fra la parte cartografica e quella gestionale.

8. CONCLUSIONI

A fronte dell'intera trattazione, le operazioni che verranno svolte dai vari uffici interessati saranno le seguenti:

1. ufficio Lavori Pubblici
    - disegno AutoCAD
    - associazione del codice identificativo a ciascun oggetto disegnato
    - esportazione file Shape
2. Ced – fase PostGIS
    - importazione in PostGIS del file Shape (tramite *MapStorer*)
    - vestizione grafica dello Shape (tramite *MapStorer*)
3. Ced – fase Web
    - pubblicazione dello Shape
    - definizione dell'accesso alle informazioni pubblicate
4. Ced – fase dblink
    - alle informazioni importate in PostGIS vengono agganciati i dati presenti nel database che censisce la rete di illuminazione comunale
5. uffici Manutenzioni(1) e Lavori Pubblici(2)
    - aggiornare il database contenente i dati sulla rete di illuminazione in base agli interventi effettuati(1)
    - gestione dei vari contratti di fornitura(2)